AF/2153
JFW

PTO/SB/21 (04-07)
Approved for use through 09/30/2007. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

Total Number of Pages in This Submission	[redacted]

Application Number	09/107,618-Conf. #8313
Filing Date	June 30, 1998
First Named Inventor	Steven M. Blumenau
Art Unit	2153
Examiner Name	D. C. Dinh
Attorney Docket Number	E0295.70066US00

ENCLOSURES (*Check all that apply*)

- [ ] Fee Transmittal Form
  - [ ] Fee Attached
- [ ] Amendment/Reply
  - [ ] After Final
  - [ ] Affidavits/declaration(s)
- [ ] Extension of Time Request
- [ ] Express Abandonment Request
- [ ] Information Disclosure Statement
- [ ] Certified Copy of Priority Document(s)
- [ ] Reply to Missing Parts/ Incomplete Application
  - [ ] Reply to Missing Parts under 37 CFR 1.52 or 1.53
- [ ] Drawing(s)
- [ ] Licensing-related Papers
- [ ] Petition
- [ ] Petition to Convert to a Provisional Application
- [ ] Power of Attorney, Revocation Change of Correspondence Address
- [ ] Terminal Disclaimer
- [ ] Request for Refund
- [ ] CD, Number of CD(s) _____
  - [ ] Landscape Table on CD
- [ ] After Allowance Communication to TC
- [ ] Appeal Communication to Board of Appeals and Interferences
- [x] Appeal Communication to TC (Response to Second Notice of Non-Compliant Appeal Brief)
- [ ] Proprietary Information
- [ ] Status Letter
- [x] Other Enclosure(s) (please Identify below):

  Amended Appeal Brief
  Return Receipt Postcard

Remarks

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm Name	WOLF, GREENFIELD & SACKS, P.C.		
Signature	[signature]		
Printed name	Richard F. Giunta		
Date	October 23, 2007	Reg. No.	36,149

Certificate of Mailing Under 37 CFR 1.8(a)

I hereby certify that this paper (along with any paper referred to as being attached or enclosed) is being deposited with the U.S. Postal Service on the date shown below with sufficient postage as First Class Mail, in an envelope addressed to: Mail Stop Appeal Brief-Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

Dated: October 24, 2007 Signature: Patricia L. Marchetti (Patricia L. Marchetti)

ATTORNEY'S DOCKET NO: **E0295.70066US00**

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	Steven M. Blumenau et al.
Serial No:	09/107,618
Confirmation No:	8313
Filed:	June 30, 1998
For:	METHOD AND APPARATUS FOR PROVIDING DATA MANAGEMENT FOR A STORAGE SYSTEM COUPLED TO A NETWORK
Examiner:	Dinh, Dung C.
Art Unit:	2153

Certificate of Mailing Under 37 CFR 1.8(a)

I hereby certify that this paper (along with any paper referred to as being attached or enclosed) is being deposited with the U.S. Postal Service on the date shown below with sufficient postage as First Class Mail, in an envelope addressed to: Mail Stop Appeal Brief - Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

Dated: October 24, 2007

Patricia L. Marchetti

Patricia L. Marchetti

RESPONSE TO SECOND NOTICE OF NON-COMPLIANT APPEAL BRIEF

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

A Second Notification of Non-Compliant Appeal Brief (Second Notification) was mailed in connection with this application on September 24, 2007, which alleged that the Appeal Brief failed to comply with the requirements of 37 C.F.R. 41.37(c)(1)(x) relating to the Related Proceedings Appendix. In response to a First Notification of Non-Compliant Appeal Brief (First Notification), Appellant indicated that the Related Proceedings Appendix consists of the statement "none." In the Second Notification, the Examiner makes note of the indication, but asserts that the Related Proceedings Appendix is not under the proper heading. In a telephone conversation, Appellant's representatives ask the Examiner whether submitting a new Appeal Brief with the appropriate amendments, or submitting a Related Proceedings Appendix on a

separate sheet without submitting a new Appeal Brief was the appropriate action in response to the Second Notification. The Examiner recommended submitting a new Appeal Brief with the appropriate amendments. Accordingly, Appellant submits a new Appeal Brief with the following amendments:

1) The Table of Contents has been updated to correct for page number changes resulting from the amendments.

2) The SUMMARY OF CLAIMED SUBJECT MATTER (37 C.F.R. §41.37(c)(1)(v)) in Part V has been amended to include a recitation of each independent claim with references to the specification, thus incorporating the subject matter submitted in response to the First Notification mailed on May 22, 2007. The response to the First Notification was filed June 22, 2007. Thus, the new Appeal Brief is believed to be in full compliance with 37 C.F.R. §41.37(c)(1)(v).

3) APPENDIX B (37 C.F.R. §41.37(c)(1)(ix)) - EVIDENCE is added and consists of the statement "NONE." Thus, the new Appeal Brief is believed to be in full compliance with 37 C.F.R. 41.37(c)(1)(ix).

4) APPENDIX C (37 C.F.R. §41.37(c)(1)(x)) - RELATED PROCEEDINGS is added and consists of the statement "NONE." Thus, the new Appeal Brief is believed to be in full compliance with 37 C.F.R. 41.37(c)(1)(x).

5) APPENDIX D now includes the Hardcopies of HTML Link Citations (previously presented as APPENDIX B).

Appellant respectfully requests that the new Appeal Brief replace the original Appeal Brief.

CONCLUSION

The new Appeal Brief is believed to be in full compliance with 37 C.F.R. §41.37. If the Examiner believes the new Appeal Brief remains non-compliant, the Examiner is respectfully requested to contact the undersigned at the telephone number provided below.

Respectfully submitted,
Steven M. Blumenau et al., Appellant

By: _____

Richard F. Giunta, Reg. No. 36,149
Wolf, Greenfield & Sacks, P.C.
600 Atlantic Avenue
Boston, Massachusetts 02210
(617) 646-8000
Attorneys for Appellant

Docket No.: E0295.70066US00
Date: October 23, 2007

DOCKET NO: E0295.70066US00

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	Steven M. Blumenau et al.
Serial No:	09/107,618
Confirmation No:	8313
Filed:	June 30, 1998
For:	METHOD AND APPARATUS FOR PROVIDING DATA MANAGEMENT FOR A STORAGE SYSTEM COUPLED TO A NETWORK
Examiner:	Strange, Aaron N.
Art Unit:	2153

Certificate of Mailing Under 37 CFR 1.8(a)

I hereby certify that this paper (along with any paper referred to as being attached or enclosed) is being deposited with the U.S. Postal Service on the date shown below with sufficient postage as First Class Mail, in an envelope addressed to: Mail Stop Appeal Brief - Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

Dated: October 24, 2007

Patricia L. Marchetti

Patricia L. Marchetti

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF

Richard F. Giunta, Reg. No. 36,149
Wolf, Greenfield & Sacks, P.C.
600 Atlantic Avenue
Boston, Massachusetts 02210
(617) 646-8000
Attorneys for Appellant

TABLE OF CONTENTS

This brief is in furtherance of the Notice of Appeal mailed on January 17, 2007.

I. REAL PARTY IN INTEREST (37 C.F.R. §41.37(c)(1)(i))

The real party in interest in this application is the assignee, EMC Corporation, a Massachusetts corporation having a place of business at 171 South Street, Hopkinton, Massachusetts 01748.

II. RELATED APPEALS AND INTERFERENCES (37 C.F.R. §41.37(c)(1)(ii))

There are no other appeals or interferences known to the Applicant, the Applicant's legal representative, or the assignee which will directly affect, be directly affected by, or have a bearing on the Board's decision in an appeal.

III. STATUS OF CLAIMS (37 C.F.R. §41.37(c)(1)(iii))

There are 32 total claims in this application (3 independent claims and 29 dependent claims). The following list summarizes the status of the claims:

1. Claims pending and appealed: 1-4, 6-27 and 29-34
2. Claims rejected: 1-4, 6-27 and 29-34
3. Claims allowed: none
4. Claims withdrawn from consideration: none
5. Claims canceled: 5 and 28

IV. STATUS OF AMENDMENTS (37 C.F.R. §41.37(c)(1)(iv))

No amendments have been filed subsequent to the final Office Action of October 18, 2006. No amendments are not entered.

V. SUMMARY OF CLAIMED SUBJECT MATTER (37 C.F.R. §41.37(c)(1)(v))

Embodiments of the present invention relate to access security in the context of a shared storage system, such as a data storage resource shared by multiple host devices connected to a network, an example of which is shown in FIG. 3 of the Appellant's specification.

Conventional shared storage systems may prevent host devices from accessing portions of storage devices for which they are not authorized by selectively allowing and denying access according to access privileges stored, for example, in a look-up-table (LUT) or database, that indicate which host devices are allowed to access which portions of the data storage device. The process of allowing and denying access based on stored privileges is referred to as *authorization*. Such conventional shared data storage systems have been implemented in the context of local networks in a trusted environment. Thus, authorization techniques are sufficient to prevent unauthorized access.

In a trusted network environment, the host devices are trusted not to obtain and use the identity of another host device to attempt to gain the access privileges of the other host device (a security threat referred to as "spoofing"), either because all of the host devices connected to the network are known, or because the network configuration itself prevents a host device from spoofing another's identity. In a typical trusted environment, an administrator or operator is responsible for connecting the host devices to the shared resource (e.g., over connections using the Small Computer System Interface (SCSI) protocol) and establishing access privileges for each of the host devices. In such an exclusive environment, security threats having to do with host devices pretending to be other host devices to hijack the access privileges of another (e.g., the access another's data) are not present.

Appellant appreciated that authorization techniques may be insufficient to prevent unauthorized access to a shared resource implemented in an untrusted network environment. In particular, as shared data storage networks expand and become more inclusive, security risks associated with untrusted host devices being connected to the network and operating as bad actors (e.g., host devices endeavoring to spoof the identify of other host devices) become an

actual and real concern. To address security risks, such as those presented by spoofing, Appellant developed techniques to confirm that a host device is indeed the host device that it represents itself to be, a process referred to as authentication or verification, to prevent spoofing or other methods of identity theft used to hijack another host device's access privileges (page 11, line 26 *et seq.*, of Appellant's specification).

In view of the foregoing, one aspect of the present invention is directed to techniques for managing access to a plurality of volumes of a storage system by at least two devices coupled to the storage system through a network. In some embodiments, the techniques comprise both authorizing and authenticating a device requesting access to the storage system. In some embodiments, the techniques may comprise determining whether a device that represents itself as a particular device and makes a request to access at least one of the plurality of volumes is authorized to access the requested volume (authorization), and verifying that the one of the requesting device is indeed the device that it represents itself to be (authentication).

The foregoing summary of the invention is provided merely to assist the Board in appreciating various aspects of the present invention. However, this summary may not apply to each of the independent claims on appeal, and the language of the independent claims may differ in material respects from the summary provided above. Thus, the Board is respectfully requested to give careful consideration to the language of each of the independent claims, which are summarized below, and to address each on its own merits, without relying on the summary provided above. In this respect, Appellants do not rely on the summary provided above to distinguish any of the claims of the present invention over the prior art, but rather, rely only upon the arguments provided in Section VII below.

A) Independent Claim 1

Claim 1 is directed to a data management method for managing access to a plurality of volumes of a storage system 20 by at least two devices 12, 14 coupled to the storage system through a network 21. The method comprises steps of receiving over the network 21 at the

storage system 20 a request from one of the at least two devices for access to at least one of the plurality of volumes of the storage system (page 6, lines 15 and 16), the request identifying the at least one of the plurality of volumes in the storage system and a represented source of the request (FIG. 2; page 7, lines 3-23). The method further comprises steps of selectively servicing the request, at the storage system, based at least in part on steps of determining, from configuration data, whether the represented source is authorized to access the at least one of the plurality of volumes (FIG. 3; page 3, line 29- page 4, line 3; page 7, lines 23-25; page 8 line 3 – page 10, line 4), and verifying that the represented source of the request is the one of the at least two devices that issued the request (FIG. 6; page 11, line 26 – page 12, line 3; page 12, line 4 – page 14, line 30).

B) Independent Claim 15

Claim 15 is directed to a computer readable medium comprising a first data structure 76 to manage accesses by a plurality of devices 12, 14, 16 to volumes of data at a storage system 20 over a communication network 21, the storage system managing access responsive to requests that each identifies one of the plurality of volumes of the storage system to be accessed and one of the plurality of devices that is represented as having issued the request (FIG. 2; page 7, lines 3-23), the first data structure 76 comprising a plurality of records 76a-76n corresponding to the plurality of devices, the plurality of records comprising at least one record corresponding to one of the plurality of devices and including configuration information having at least one identifier that identifies which of the volumes of the storage system the one of the plurality of devices is authorized to access (page 9, lines 15-29), and authentication information that can be used by the storage system to determine whether the one of the plurality of devices that issued the request is the corresponding one of the plurality of devices (FIG. 6; page 11, line 26 – page 12, line 3; page 12, line 4 – page 14, line 30).

C) Independent Claim 21

Claim 21 is directed to a storage system 20 comprising at least one storage device 38 apportioned into a plurality of volumes, a configuration table 32 to store configuration data identifying which of a plurality of devices 12, 14, 16 coupled to the storage system via a network are authorized to access which of the plurality of volumes (FIG. 3; page 3, line 29 – page 4, line 3; page 7, lines 23-25; page 8 line 3 – page 10, line 4), and a filter 34, responsive to the configuration data, to selectively forward to the at least one storage device requests for access to the plurality of volumes received from the plurality of devices over the network (page 9, line 30 – page 10, line 28), wherein each request identifies at least one of the plurality of devices that is represented to the storage system as having issued the request (FIG. 2; page 7, lines 3-23), and wherein the filter 34 is adapted to verify that the at least one of the plurality of devices 12, 14, 16 identified in the request is the device that issued the request (FIG. 6; page 11, line 26 – page 12, line 3; page 12, line 4 – page 14, line 30).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL (37 C.F.R.§41.37(c)(1)(vi))

1. Whether claims 1-4, 9-27, 29-32 are obvious under 35 U.S.C. §103 over U.S. Patent No. 6,061,753 (Ericson) in view of U.S. Patent No. 4,919,545 (Yu) and U.S. Patent No. 5,959,994 (Boggs).

VII. ARGUMENT (37 C.F.R. §41.37(c)(1)(vii))

1. Claims 1-4, 9-27 and 29-32 are Not Obvious Under §103 Over Ericson in View of Yu and Boggs

Each of the above-listed claims stands rejected under 35 U.S.C. 103(a) as purportedly being obvious over Ericson in view of Yu and Boggs by the final Office Action mailed October 18, 2006 (the 10/18 Office Action). Initially, Appellant points out that the 10/18 Office Action asserts that the combination of Ericson and Yu discloses each of the limitations of the above-listed claims, relying on Boggs merely as evidence that purportedly supports the allegation that the combination of Ericson and Yu is proper. The 10/18 Office Action does not apply Boggs to provide any disclosure absent in the alleged combination of Ericson and Yu with respect to the claims. Before turning to Appellant's arguments with respect to the improper combination of Ericson and Yu, a summary will be provided of the teachings of each of Ericson and Yu, followed by a brief discussion of trusted and untrusted network environments.

A. Ericson

Ericson is directed to controlling access to a target device 102 (e.g., a disk array) by initiators 100 interconnected by a small computer system interface bus ("SCSI bus") 104, wherein the network devices are preconfigured in accordance with the SCSI specification. (Col. 3, lines 48-56). FIG. 1 of Ericson is reproduced below.

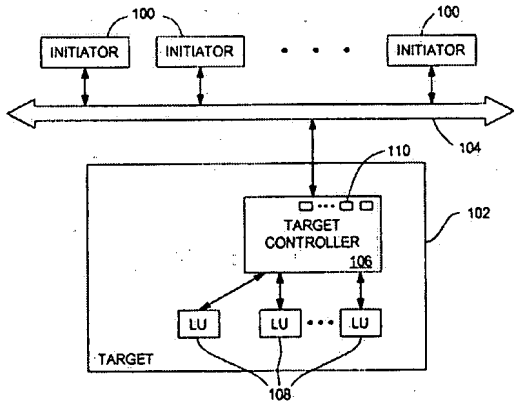


FIG. 1

The initiators 100 request access to the target 102 by directing an access message to the target 102 via the SCSI bus 104, the message including the initiator identifier, a target identifier and a portion of the target device 102 to be accessed (i.e., the logical unit 108). (Col. 3, lines 56-61). Access to the target device is controlled by a look-up structure preconfigured by a system operator who assigns selected logical units 108 in the target 102 to each of the initiators 100. When an initiator requests access, the look-up structure is indexed to check whether the initiator has been authorized to access the logical unit identified in the access message before allowing the initiator to access the logical unit (Col. 2, lines 26-65). Ericson is completely silent with respect to verifying that an initiator matches the initiator identifier in the message, or any other authentication techniques. Nor does Ericson anywhere mention anything about security breaches of this nature, as discussed in further detail below.

B. Yu

Yu is directed to distributed security measures in an intelligent network having a plurality of interconnected network nodes, each having specified resources (e.g., a distributed telecommunication network architecture). FIG. 1 of Yu is reproduced below.

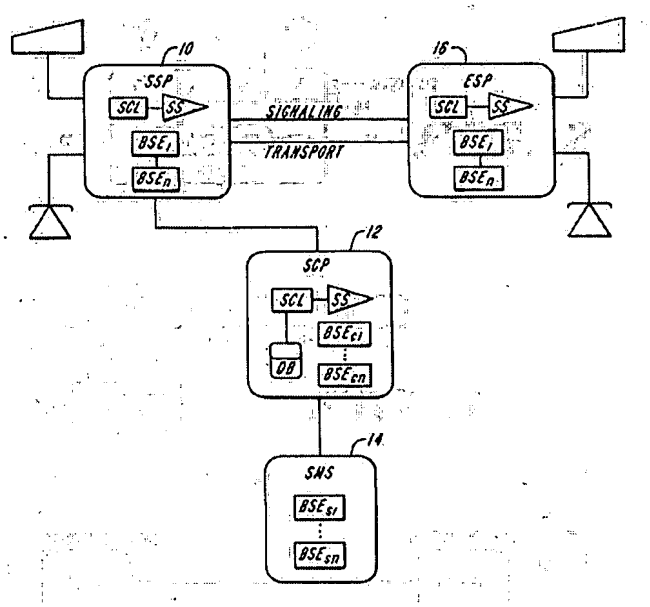


FIG. 1

The distributed network service architecture shown in FIG. 1 includes a plurality of interconnected nodes 10, 12, 14 and 16. Node 10 is service switching point (SSP) adapted to process a telephone call (col. 3, line 67 – col. 4, line 1). Node 12 is a service control point (SCP) database containing service control logic (SCL), service scripts (SS) and customer specific information (col. 3, lines 1-4). Node 14 is a service management system (SMS) which operates as an administration and control system supplying operations for the creation of service scripts, for updates to the database in node 12, and for the support of customer control over services (col. 3, lines 4-9). Service switching point 10 is also connected to node 16, which is an enhanced service provider (ESP) connected outside the network. The enhanced service provider at node

16 can utilize basic service elements provided by the service switching points. (Col. 3, lines 9-14).

The architecture in FIG. 1 allows outside service providers to access service elements of the telephone network (col. 3, lines 37-39). In particular, the various nodes store objects associated with network resources having sets of interface operations describing the services provided by the respective object. Nodes can access the network resources of an object through the invocation of the service elements of the interface of the corresponding object. (Col. 5, lines 1-11). Objects can be identified and protected using a mechanism called a capability. A capability is a unique identifier of an object and a permission that gives a node the right to access the object. When a node requests a service provided by an object residing at another node (referred to as the execution node), a capability for the object is returned to the requesting node, giving it access to the services in the object permitted by the capability. (Col. 5, line 62 – Col. 6, line 11).

With the intelligent network architecture, outside service providers can access service elements of the telephone network. Yu teaches that in a distributed architecture, protection mechanisms for network security and integrity should themselves be distributed, but that if so they are vulnerable to forgery, theft, modification or destruction. (Col. 4, lines 37-65). Because an intruder in such a distributed network can forge almost all parts of a transmitted message except the node address (e.g., the network interface hardware or other low level control), Yu teaches that unique encryption keys be exchanged between an execution node and a node requesting a capability from the execution node, based on the low level hardware address (Col. 6 line 34 – col. 8 line 60). If a capability cannot be decrypted using the unique pairing of keys, the execution node knows that the capability has been tampered with, or an imposter node transmitted the capability of another node. (Col. 7 line 45 – col. 8 line 60).

C. Trusted and Untrusted Environments

From a security standpoint, the networks described in Ericson and Yu are very different and therefore exhibit different security concerns. In particular, as discussed below, Ericson describes a networked data storage system operating in a trusted environment. Yu describes a network for providing various services in an untrusted environment. A trusted network environment is one wherein devices seeking access to a resource are trusted not to spoof (or are incapable of spoofing) another's identity to hijack the access permissions allocated to another device. In such an environment, it is sufficient to allocate access privileges to each device that selectively allow and prevent devices from accessing portions of a resource according to the allocated access privileges. As discussed above, this security measure is referred to as authorization.

In an untrusted network, the network may be widely distributed and the devices on the network may be unknown to the owner of the resource, making the network vulnerable to intruders or other bad actors (see e.g., Yu, cols. 8 and 9). Accordingly, authorization may not be sufficient because a request representing itself as being from a particular device cannot be trusted to have genuinely been sent from that device rather than from another device attempting to spoof its identity to gain unauthorized access to information. Accordingly, Yu implements authentication techniques to verify that a device is actually the device it represents itself to be, thus preventing unauthorized access to a resource via spoofing or capability theft.

It should be appreciated from the foregoing that *authorization* and *authentication* are two distinct security measures that address separate and distinct security issues. Authentication techniques are unnecessary in a trusted environment and do not serve to increase security, but rather add unnecessary complexity to the access management scheme in a trusted environment. That is, authentication techniques implemented in a trusted environment serve only to protect against non-existent security threats.

D. The Final Office Action Fails to Establish a Prima Facie Case of Obviousness

MPEP §2142 states, "[t]o establish a *prima facie* case of obviousness...there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings." The legal concept of *prima facie* obviousness allocates the initial burden of producing factual support to the Examiner (MPEP §2142). In particular, the Examiner bears the initial burden of establishing that there is some suggestion or motivation to combine the references.

The Examiner has failed to produce evidence from the references or elsewhere to support the allegation that there is motivation to combine Ericson and Yu. In the "Response to Arguments" section in the final Office Action mailed on September 19, 2005 (9/19 Office Action), the 9/19 Office Action asserts that "both Ericson and Yu are directed to improving access security, hence they are analogous art." During a telephone conference conducted on November 15, 2005, the Examiner re-asserted this position, further alleging that since both Ericson and Yu involve access security in a network environment, it would have been obvious to use the authentication measures described in Yu to improve the access security of Ericson. However, even if it is true that Ericson and Yu are in the same field, that alone does not prove motivation to combine the references.

The Examiner must show incentive to combine the teachings of the two references. The MPEP §2142, third paragraph of the section entitled "ESTABLISHING A *PRIMA FACIE* CASE OF OBVIOUSNESS," cites *Ex parte Skinner* (Decision of the Board of Patent Appeals and Interferences,. No. 650-69), which states that "[w]hen the incentive to combine the teachings of the references is not readily apparent, it is the duty of the examiner to explain why the combining of the reference teachings is proper." The Office Action fails to provide any evidence that Yu's authentication methods actually *would* improve the security of Ericson. Not all networks are the same, nor do they exhibit the same security concerns. A security measure taken in one

environment may be entirely irrelevant in another, and may not improve security, but rather may be an unnecessary design implementation.

In the "Response to Arguments" section of the final Office Action mailed on April 21, 2006 (the 4/21 Office Action), the final Office Action notes that Ericson reports that other protocols such as Fibre Channel may be used to implement the invention disclosed in Ericson. The 4/21 Office Action asserts that the mention of the Fibre Channel protocol is evidence that Ericson describes a system for use in untrusted networks, and therefore the authentication methods of Yu would be expected to improve the security in Ericson. However, the mere mention of a network protocol that is sufficiently generic that it could be used in other system configurations that are untrusted does not somehow override Ericson's teaching (discussed further below) that the disclosed system is intended for use in a trusted environment wherein identity spoofing is not a concern. Accordingly, the contention that the authentication methods of Yu would have somehow improved the security of the network storage system of Ericson is entirely unsupported.

In a Response to the Office Action mailed on September 19, 2005 (the 9/19 Office Action), Appellant argued that the 9/19 Office Action did not overcome the Examiner's burden to "explain why the combining of the reference teachings is proper," as required by *Ex Parte Skinner*. In response, the Examiner cited Boggs in the 4/21 Office Action as evidence that allegedly supports the assertion that the combination of Ericson and Yu is proper. Since the Examiner cannot find the motivation that would compel one of ordinary skill in the art to combine Ericson and Yu in the references themselves, it is assumed that the Examiner is using Boggs to demonstrate the "knowledge generally available to one of ordinary skill in the art," as required by MPEP §2142. However, Boggs does not provide the requisite motivation to combine Ericson and Yu, as discussed in further detail below.

i. Boggs Does Not Provide Motivation to Combine Ericson and Yu

The 4/21 Office Action and the 10/18 Office Action reference Boggs in a number of locations, the 10/18 Office Action being substantially a restatement of the 4/21 Office Action.

The 10/18 Office Action asserts on page 3 in the "Response to Arguments" section that "Boggs teaches that Fibre Channel is considered to be preferable to parallel bus SCSI ..., providing motivation to use the Fibre Channel embodiment suggested by Ericson." Appellant respectfully points out that motivation to use the "Fibre Channel embodiment" of Ericson is not what is needed to establish a *prima facie* case of obviousness. Rather, what is required, is motivation to use Yu's authentication techniques on Ericson's disclosed system, which Boggs simply does not provide. The 10/18 Office Action nowhere explains how Boggs provides motivation to combine Ericson and Yu.

In fact, the only relevant motivation to combine Ericson and Yu asserted by the Examiner is taken from Appellant's own specification. In particular, the 10/18 Office Action states on page 3 that "[a]s acknowledged by Applicant, it is known that Fibre Channel environments are not always trusted." Thus, the Examiner concludes, "one of ordinary skill in the art, when presented with the disclosures of Ericson and Boggs, and aware of the known security issues present in a Fibre Channel environment, would have been motivated to seek out solutions to them." Ericson and Boggs disclose absolutely nothing about security issues present in untrusted network environments, nor describe Fibre Channel in the context of untrusted network environments. The Examiner's sole motivation to combine is obtained from Appellant's own specification. That is, Appellant alone provides the insight that shared storage systems implemented in untrusted environments may need additional security measures. Neither Ericson nor Boggs mentions anything of the sort.

On page 4 of the 10/18 Office Action, the Examiner states that it "is well known in the art at the time of the invention that SCSI peripherals may be distributed over wide area network using ATM and Fibre Channel," citing Boggs at col.2, lines 63-68 and col. 10 lines 8-22 to support this assertion. Appellant disagrees with both the assertion and the conclusions drawn from it. In particular: 1) Boggs, and more particularly, the excerpts cited by the Examiner do not teach SCSI peripherals distributed over a wide area network using Fibre Channel; and 2) even if this assertion were true, such a teaching indicates nothing about whether the systems

described in Ericson are untrusted, and therefore has no bearing on motivation to combine Ericson and Yu.

With respect to 1) above, to the extent that Boggs teaches the use of ATM to communicate with the disk arrays 122-126 (Fig. 1), it teaches that ATM should replace the use of the SCSI bus to communicate between the disk arrays and processing systems. Referring specifically to the passages cited in the 10/18 Office Action, the first passage (at col. 2, lines 63-68) says nothing more than that Fibre Channel is becoming a more popular peripheral interconnection technology than the parallel SCSI bus. This indicates nothing about whether the system of Ericson contemplates untrusted environments. As discussed above, this disclosure provides, at best, motivation to use the Fibre Channel protocol, which is irrelevant to the alleged motivation to modify Ericson to include authentication.

Similarly, the disclosure at col. 10, lines 8-22 indicates that ATM is a preferred universal interconnect for peripherals, and that it can be adapted to use the SCSI Fibre Channel protocol (FCP), which is the serial SCSI protocol supported by Fibre Channel. The fact that the Fibre Channel standard can be adapted to support the SCSI protocol provides nothing at all suggesting that the trusted environment of Ericson is somehow untrusted. As discussed in further detail below, there is nothing inherently untrusted about the Fibre Channel protocol that would convert a trusted environment to an untrusted environment merely by the implementation of the protocol.

The only disclosure in Boggs related to systems employing a SCSI bus or implementing the Fibre Channel protocol is found on column 2, lines 27-57 describing the systems in Figs. 11-13. In the above identified disclosure, a configuration is shown and described wherein a number of disk arrays 1122-1128/1222-1228 are connected to a number of processing systems 1102-1108/1202-1208 via a SCSI bus 1130/1230. The processing systems are connected via a local area network (LAN) 1122/1212 to clients. It should be appreciated, however, that the communication between the clients and the processing systems 1102-1108/1202-1208 does not occur over the SCSI bus 1130/1230. That is, the processing clients communicate with outside LAN clients over the LAN cloud 1122/1212 and not via the SCSI bus. The SCSI bus is used

only for access to the disk arrays from the processing systems. Thus, to the extent that Boggs discloses anything about SCSI or a Fibre Channel implementation over SCSI, Boggs teaches a standard trusted SCSI network. Boggs is completely silent with respect to untrusted network environments, or security risks that may be present in untrusted network environments.

With respect to 2), even if Boggs did disclose that, such disclosure would have no bearing on whether Ericson contemplates untrusted networks. Fibre Channel is merely a protocol that can be implemented as a peripheral interface in a variety of environments, some of which may be trusted and some of which may be untrusted. There is nothing intrinsically untrusted about the Fibre Channel protocol. In fact, that the Fibre Channel protocol can be implemented in untrusted environments is not in dispute. As such, even a Prior Art reference teaching an untrusted environment on which a Fibre Channel protocol has been implemented (which has not been provided) would have no bearing on whether the alleged "Fibre Channel embodiment" in Ericson would be trusted or untrusted. In particular, the only disclosure related to the "Fibre Channel embodiment" is the found in column 6, lines 1-6 of Ericson, which states:

> Although embodiments of the invention have been described in terms of the SCSI specification, it should be noted that other communications protocols may be utilized to implement the invention. For example, conventionally known peripheral connect interface ("PCI") and Fibre Channel protocols may be utilized.

Ericson does not describe an alternate system or environment for such embodiments. Accordingly, nothing can be said about the nature of the "Fibre Channel embodiment" except that it is implemented using the Fibre Channel protocol, which is not inconsistent with trusted environments. Implementation of the Fibre Channel protocol does not by itself convert a trusted environment into an untrusted environment. Accordingly, for the reasons provided in the foregoing, Boggs provides no basis for the allegation that Ericson contemplates an untrusted network environment, and therefore does not provide motivation to combine Ericson and Yu, or otherwise suggest why one of ordinary skill in the art would have been motivated to modify the

trusted system of Ericson with authentication techniques relevant only to untrusted environments.

To establish a *prima facie* case of obviousness, it must be shown *why* one skilled in the art, viewing the references at the time of the invention, would have been motivated to add Yu's authentication into Ericson (*Ex parte Skinner*). Accordingly, the Office Action has the burden of showing why one skilled in art would have believed that authentication would have improved security in the Ericson system. Merely stating that Yu's authentication would have improved the security of Ericson without factual support from the references is not sufficient to meet the Office Action's burden. Furthermore, Boggs provides no evidence to support the assertion that Ericson contemplates untrusted environments. Accordingly, the Office Action has not established a *prima facie* case of obviousness, and the rejection is therefore improper.

E. There is No Motivation to Combine Ericson and Yu.

As discussed above, the Office Action has failed to meet the burden of showing motivation to modify Ericson based on Yu. Therefore, Applicant need not produce evidence to the contrary (MPEP §2142). However, even though Applicant is not obligated to show why it would not have been obvious to combine Ericson and Yu, Applicant provides below support from the references showing that the authentication techniques of Yu would not in fact improve the security of Ericson, so that one skilled in the art would not have been motivated to modify Ericson as alleged in the Office Action because it would have merely complicated the system with unnecessary measures that do not improve security.

Nowhere does Ericson describe untrusted environments or any type of security breaches requiring authentication, and the Office Action points to no such teaching. Rather, the Office Action points out that Ericson mentions that "conventionally known peripheral connection interfaces ("PCI") and Fibre Channel protocols may be utilized" (col. 6, lines 4-6), and asserts that this is a teaching that Ericson teaches the use of the disclosed techniques in an untrusted environment. However, that is an entirely unsupported leap from the mere mention that the Ericson authorization mechanism can be implemented using a particular protocol. Ericson

mentions Fibre Channel merely to illustrate that the invention can be implemented using different protocols.

Fibre Channel can be used to implement networks in a trusted environment and the Office Action does not even attempt to allege otherwise. The 4/21 and 10/18 Office Actions cite Boggs for the proposition that Fibre Channel can be used to implement untrusted networks. As discussed above, Boggs does not in fact disclose an untrusted network environment, as Boggs is completely silent with respect to any security threats, and in particular, spoofing risks that are the hallmarks of an untrusted network. In fact, in the context of shared data storage, the only network environments disclosed by Boggs are implemented by a SCSI bus, which are inherently trusted, as discussed below. Thus, the Office Action's assertion that the mention of the Fibre Channel protocol (which is entirely consistent with a trusted environment) in connection with a reference (Boggs) that discusses Fibre Channel without any mention of security or untrusted environments somehow provides a teaching that the Ericson system be used in an untrusted environment so that security measures of the type used in Yu should be considered is entirely unsupported.

Nothing is mentioned to suggest that the Eriscon system, even if implemented using the Fibre Channel protocol, would be used in an untrusted environment. Ericson (and Boggs) are completely silent with respect to security risks in widespread networks, the dangers of environments that are vulnerable to spoofing, or any other disclosure that would have motivated one skilled in the art to seek security measures that would be useful in an untrusted network environment but entirely unnecessary in a trusted environment. Furthermore, not only is Ericson silent about an untrusted environment, but Ericson's entire description is in the context of a trusted environment. In particular, the controlled data storage access system of Ericson is performed in a SCSI environment where initiators are trusted. As discussed below, both the nature of the SCSI environment and the details of the SCSI interface make it unnecessary and therefore undesirable to implement verification or authentication methods as disclosed by Yu.

The SCSI protocol defines a standard for communication between a computer and various peripheral devices. In particular, various host devices (referred to as initiators) may issue requests to one or more peripheral devices (referred to as targets) that are connected to the SCSI bus. Each device (i.e., an initiator or a target) has a unique SCSI ID which identifies its physical location on the SCSI bus. The narrow SCSI bus and associated connectors support a maximum of eight devices. The wide SCSI bus and associated connectors support a maximum of 16 devices. See e.g., http://scsifaq.paralan.com/ and, in particular, http://scsifaq.paralan.com/scsifaqanswers.html#9 and http://scsifaq.paralan.com/scsifaqanswers.html#10.

The SCSI environment is local and contained, and therefore trusted and secure. Only a limited number of devices can be attached to a SCSI bus over a limited and local area. For example, internal SCSI connections (i.e., SCSI ribbon connections) are designed for communication between components and peripherals within the same computer (e.g., a personal computer). See e.g., http://computer.howstuffworks.com/scsi3.htm. External SCSI connections (i.e., SCSI cables) are designed to connect peripherals over relatively short distances. For example, SCSI cables typically come in 3ft and 6ft lengths. Although the cables may be daisy-chained, the SCSI protocol itself does not support bus lengths greater than 25 meters. Accordingly, a SCSI network is limited to a small area and limited to a small number of devices. See e.g., http://www.ramelectronics.net/html/scsi_connecters.html#cablelength.

In addition, each device on the SCSI bus must be manually configured and physically connected to the bus via an appropriate SCSI connector, and assigned a unique SCSI ID (often by manually setting a physical switch or configuring external jumpers). To assign a unique SCSI ID to a device (i.e., 0-7 for narrow SCSI and 0-15 for wide SCSI), the SCSI ID of every other device on the bus must be known to avoid conflicts. See e.g., http://support.gateway.com/s/CDROM/Panasonc/CS006aa/PANAS100.shtml.

Accordingly, there are no unknown or untrusted devices connected to a SCSI bus. An operator or administrator building a SCSI network must physically attach each device to the bus

and configure it appropriately. http://www.sun.com/solutions/blueprints/0800/scsi.pdf (see especially "SCSI Issues in Clusters" on page 3, *et. seq.*) Therefore, the operator is cognizant of each device on the network and is fully in control of what devices are connected to the SCSI bus. That is, untrusted devices cannot gain access to the SCSI bus. In such a local and trusted environment, it would have been unnecessary to implement verification and/or authentification procedures.

Furthermore, the SCSI interface itself prevents a device from misrepresenting its identity. The SCSI ID assigned to each device connected to a SCSI bus both identifies the device and specifies the device's physical address on the bus. The uniqueness of a SCSI ID is a requirement of the interface. http://www.sun.com/solutions/blueprints/0800/scsi.pdf. For example, bus arbitration and communication depends on each attached device having a unique SCSI ID. Conflicts with SCSI IDs prevent the conflicting devices from gaining access and communicating over the SCSI bus (and may result in the failure of all devices on the SCSI bus). http://www.ba-stuttgart.de/~schulte/htme/ebuss12.htm#REF2.1.2. The uniqueness of a device's SCSI ID is its license to access the bus. For a device to communicate over the bus, it must represent itself by that unique SCSI ID. Accordingly, there is no way for a device to misrepresent itself without disrupting the SCSI network.

In Ericson, a plurality of initiators 100 are connected via a SCSI bus 104 to a target device 102 such as a disk array having a controller 106 (col. 3, line 53 – col. 4, line 5). Upon request by an initiator, the controller accesses a look-up data structure that defines which initiators have access to which logical units of the disk array to ensure that the request is permitted (col. 4, lines 6-54). In Ericson, the allocation of logical units to particular initiators is conducted in a trusted environment. For example, column 4, lines 54-61 state:

> The look-up data structure may be pre-configured by a system operator who assigns selected logical units 108 in the target 102 to each of the initiators 100. This preconfiguration preferably is performed when the target controller 106 is installed. When necessary, however, the look-up data structure may be reconfigured at any subsequent time, such as when new initiators 100 are added

> to the system, or when the logical units 108 must be reassigned to other initiators 100.

The system operator is in complete control of the local SCSI network. The system operator configures the look-up data structure according to the devices on the SCSI bus and allocates logical units to new devices added onto the SCSI bus as desired. The system operator is trusted with properly adding devices to the SCSI bus and defining the look-up data structure to permit access as desired by associating initiator IDs with desired logical units. In a SCSI environment, the system operator must give each device a unique SCSI ID which must remain unique in order for a device to communicate on the bus.

In this environment, there is no opportunity for a device to misrepresent its identity to gain access to restricted logical units of the target device. The SCSI network is a local and contained network of devices wherein the administrator of the network has control over connecting each of the devices, configuring their SCSI ID's and allocating their respective permissions. Not only are there no untrusted devices on the SCSI network, but the SCSI network itself prevents devices from spoofing their identity to gain access credentials of another device.

While Ericson mentions that other peripheral interfaces (e.g., Fibre Channel) may be used, Ericson does not contemplate untrusted environments or anywhere suggest that the described access method could be used in environments where there is a possibility that initiators may attempt to misrepresent their identity to gain access to restricted logical units. In particular, nowhere does Ericson disclose an environment where untrusted devices have access to the data storage, nor does Ericson suggest that spoofing is, or would be, a problem. Accordingly, authenticating the identity of a device is completely unnecessary in Ericson.

F. Summary of Arguments Regarding Ericson and Yu

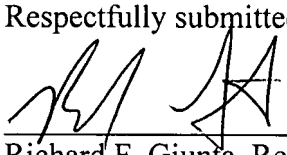
In the "Response to Arguments" section, the 9/19/05 Office Action asserts on page 2 that the motivation to combine Ericson and Yu is that Ericson benefits from "the advantage of [Yu's] security method." As discussed above, the Office Action has the burden of showing that authentication is in fact an advantage in Ericson. In order for the Office Action's assertion to operate as valid motivation to combine Ericson and Yu, the Office Action must demonstrate that there is a threat in Ericson that would justify adding authentication security measures. Absent a security threat that would be remedied by authentication, one of ordinary skill in the art simply would not have been motivated to add unnecessary complexity and expense. Analogously, no one would be motivated to purchase snow tires for their car in Southern California. It is senseless to protect against non-existent threats.

It is the Examiner's burden to prove that a threat exists in Ericson that would have motivated one skilled in the art to add Yu's authentication. The Examiner has failed to produce any such evidence, let alone evidence to rebut Appellant's evidence that there is in fact no security threat in Ericson that would be remedied by Yu's authentication. Therefore, the Examiner has failed to establish a *prima facie* case of obviousness. Accordingly, Applicant respectfully requests that the rejection of claims 1-4, 9-27 and 29-32 under 35 U.S.C. 103(a) be withdrawn.

VIII. CONCLUSION

For the foregoing reasons, the rejections of claims 1-4, 6-27 and 29-34 under 35 U.S.C. §103 are improper and should be reversed.

Respectfully submitted,



Richard F. Giunta, Reg. No. 36,149
WOLF, GREENFIELD & SACKS, P.C.
600 Atlantic Avenue
Boston, MA 02210
Tel: (617) 646-8000
Attorneys for Appellants

Attorney Docket No.: E0295.70066US00
Dated: October 23, 2007

APPENDIX A – CLAIMS AS PENDING

1. A data management method for managing access to a plurality of volumes of a storage system by at least two devices coupled to the storage system through a network, the method comprising steps of:

receiving over the network at the storage system a request from one of the at least two devices for access to at least one of the plurality of volumes of the storage system, the request identifying the at least one of the plurality of volumes in the storage system and a represented source of the request; and

selectively servicing the request, at the storage system, based at least in part on steps of:

determining, from configuration data, whether the represented source is authorized to access the at least one of the plurality of volumes; and

verifying that the represented source of the request is the one of the at least two devices that issued the request.

2. The data management method according to claim 1, wherein the configuration data is stored in the storage system in a configuration table comprising a plurality of records, each of the records including an identifier and information indicating which of the volumes of data are available to a device associated with the corresponding identifier, and wherein the step of selectively servicing further includes steps of:

receiving the request at the storage system issued by the one of the at least two devices, the request including a source identifier identifying the one of at the at least two devices that initiated the request and an address to one of the volumes of the plurality of volumes in the storage system; and

determining whether to service the request responsive at least to a portion of the configuration data associated with the source identifier and the address of the one of the volumes.

3. The data management method according to claim 1, the method including a step of:

forwarding the request from the one of the at least two devices to the storage system over the network.

4. The data management method according to claim 3, wherein the step of forwarding includes forwarding the request using a Fibre Channel protocol.

5. (Canceled)

6. The data management method according to claim 33, wherein the act of verifying includes an act of verifying that the represented source of the request is the one of the at least two devices that issued the request based, at least in part, on a comparison between the request access key and the expected access key.

7. The data management method according to claim 6, wherein the request access key is encrypted using a key associated with the one of the at least two devices that issued the request.

8. The data management method according to claim 7, wherein the step of verifying further comprises a step of:

decrypting the request access key at the storage system using a decryption key associated with and initially provided by the one of the at least two devices identified in the request.

9. The data management method according to claim 1, wherein the one of the at least two devices is a host processor, and wherein the step of forwarding includes the step of forwarding the request from the host processor to the storage system.

10. The data management method according to claim 1, wherein at least one of the at least two devices is a file server and wherein the step of forwarding includes the step of forwarding the request from the file server to the storage system.

11. The data management method according to claim 1, wherein the storage system includes a plurality of disk drives, and wherein the step of selectively servicing includes the step of forwarding the request to one of the plurality of disk drives.

12. The data management method according to claim 1, further comprising a step of:

validating the request from the one of the at least two devices at the storage system to verify that the request was not altered during transit.

13. The data management method according to claim 2, wherein the configuration table comprises a plurality of records arranged in an array including a plurality of rows corresponding to a number of volumes of data available at the storage system and a plurality of columns corresponding to a number of ports available at the storage system, and wherein each of the records includes a bitmap having a bit corresponding to each device authorized to access each of the corresponding ports, and wherein the step of determining whether to service the request comprises steps of:

indexing the configuration database using the address provided in the request to identify an indexed record; and

comparing the bitmap of the indexed record with the source identifier to determine whether a bit of the bitmap associated with the source identifier indicates that the one of the at

least two devices associated with the source identifier has access to the volume of the storage system associated with the indexed record.

14. The data management method according to claim 1, wherein the step of selectively servicing further comprises steps of:

servicing a first request issued by a first one of the at least two devices for access to a first portion of data in the storage system responsive to configuration data associated with the first one of the at least two devices and an address of the first portion of data specified the first request; and

precluding a second request issued by a second one of the at least two devices for access to the first portion of data in the storage system from being serviced responsive to configuration data associated with the second one of the at least two devices and the address of the first portion of data specified in the second request.

15. A computer readable medium comprising:

a first data structure to manage accesses by a plurality of devices to volumes of data at a storage system over a communication network, the storage system managing access responsive to requests that each identifies one of the plurality of volumes of the storage system to be accessed and one of the plurality of devices that is represented as having issued the request, the first data structure comprising a plurality of records corresponding to the plurality of devices, the plurality of records comprising at least one record corresponding to one of the plurality of devices and including configuration information having at least one identifier that identifies which of the volumes of the storage system the one of the plurality of devices is authorized to access, and authentication information that can be used by the storage system to determine whether the one of the plurality of devices that issued the request is the corresponding one of the plurality of devices.

16. The computer readable medium according to claim 15, in combination with the storage system, wherein the computer readable medium is a memory of the storage system.

17. The combination according to claim 16, in further combination with the plurality of devices and the communication network, wherein the storage system and the plurality of devices are coupled to communicate over the communication network.

18. The combination of claim 17, wherein the storage system and the plurality of devices communicate according to a Fibre Channel network protocol.

19. The combination according to claim 16, wherein the storage system further comprises:

a second data structure comprising a plurality of records that form a copy of a subset of the plurality of records in the first data structure, wherein the subset of the plurality of records in the second data structure is associated with a subset of the plurality of devices that are logged into the storage system.

20. The combination according to claim 19, wherein the second data structure further comprises:

an array of records having a plurality of columns corresponding to the volumes of data at the storage system and a plurality of rows corresponding to a plurality of ports of the storage system, each record in the array including at least one bit corresponding to each of the plurality of devices.

21. (Previously presented) A storage system comprising:

at least one storage device apportioned into a plurality of volumes;

a configuration table to store configuration data identifying which of a plurality of devices coupled to the storage system via a network are authorized to access which of the plurality of volumes; and

a filter, responsive to the configuration data, to selectively forward to the at least one storage device requests for access to the plurality of volumes received from the plurality of devices over the network, wherein each request identifies at least one of the plurality of devices that is represented to the storage system as having issued the request, and wherein the filter is adapted to verify that the at least one of the plurality of devices identified in the request is the device that issued the request.

22. (Original) The storage system according to claim 21, wherein the filter forwards a request to a volume for servicing by the storage system responsive to the configuration data indicating that the one of the plurality of devices that issued the request is authorized to access the volume.

23. (Original) The storage system according to claim 21, wherein the filter precludes a request to a volume from being serviced by the storage system responsive to the configuration data indicating that the one of the plurality of devices that issued the request is not authorized to access the volume.

24. (Previously presented) The storage system according to claim 21, wherein the configuration table comprises a number of records, each record including an identifier and a map, the map indicating which volumes of the storage system are capable of being accessed by a device associated with the identifier, wherein each request received at the filter includes a source identifier identifying the one of the plurality of devices that issued the request and an address to one of the plurality of volumes, and wherein the filter further comprises:

a comparator to compare each request against the information in a selected record in the configuration table associated with the request to determine whether the one of the plurality of devices that issued the request is authorized to access the volume.

25. (Original) The storage system according to claim 24, wherein an identifier in the selected record corresponds to the source identifier of the request.

26. (Previously presented) The storage system according to claim 21, in combination with the plurality of devices and wherein the network couples the storage system to the plurality of devices.

27. (Original) The combination of claim 26, wherein the storage system and the plurality of devices communicate over the network using a Fibre Channel network protocol.

28. (Canceled)

29. (Original) The storage system according to claim 21, further comprising:

means for validating a request received at the storage system to verify that the request was not altered in transit.

30. (Original) The storage system according to claim 21, wherein the at least one storage device includes a plurality of disk drives.

31. (Original) The combination according to claim 26, wherein at least one of the plurality of devices is a host processor.

32. (Original) The combination according to claim 26, wherein one of the plurality of devices is a file server.

33. (Previously presented) The data management method of claim 1, further comprising an act of transferring an expected access key between the storage system and the at least one of the at least two devices, and wherein the act receiving the request includes an act of receiving a request from one of the at least two devices for access to at least one of the plurality of volumes, the request including a request access key, and wherein the act of verifying includes an act of comparing the request access key and the expected access key.

34. (Previously presented) The data management method of claim 6, further comprising an act of transferring encryption information between the storage system and the at least one of the at least two devices, and wherein the expected access key and/or the request access key are encrypted using the encryption information.

APPENDIX B (37 C.F.R. §41.37(c)(1)(ix)) — EVIDENCE

NONE

APPENDIX C (37 C.F.R. §41.37(c)(1)(x)) — RELATED PROCEEDINGS

NONE

APPENDIX D – HARD COPIES OF HTML LINKS

SCSI FAQ of frequently asked questions and answers, includes information on LVD/MSE, HVD and SE. Also see our **SCSI Glossary**

SCSI FAQs

Need help locating your SCSI question or answer? Use the search engine below. Hint: one word searches are best.

Search SCSI FAQ

Part 1 Q. | Part 1 A. | Part 2 Q.| Part 2 A. | Part 3 Q. | Part 3 A. | Part 4 Q.| Part 4 A.

SCSI FAQ PART 1

SCSI FAQ from Paralan

SCSI FAQ from Paralan

Part 1 Q. | Part 1 A. | Part 2 Q.| Part 2 A. Part 3 Q. | Part 3 A. | Part 4 Q.| Part 4 A.

SCSI FAQ PART 2

- Q. 27. I recently purchased a new SCSI drive with an SCA (or SCA-2) connector. How do I connect this to my cabled SCSI system?
- Q. 28. What is QAS?
- Q. 29. What is SCSI "packetization" (also called information units)?
- Q. 30. What are the five optional features of Ultra 160 (Ultra 3) SCSI?
- Q. 31. All the new drives have the LVD/MSE interface. Can I use them on a narrow single-ended SCSI bus?
- Q. 32. I know I can place a LVD/MSE peripheral on a single-ended bus, but the LVD interface is wide (34 pair cables) and the single-ended bus is narrow (25 pair cables). Any problems here?
- Q. 33. Can I attach narrow devices to a wide SCSI bus?
- Q. 34. I have heard the term "idc" applied to internal SCSI ribbon cables. What does it mean?
- Q. 35. Can I attach wide peripherals to a narrow SCSI bus?
- Q. 36. I note that in several of your FAQs you recommend the use of 68 pin to 50 pin adapters. Aren't they "impedance lumps" in the SCSI transmission line that can cause reflections and associated problems?

SCSI FAQ from Paralan

- Q. 37. I had a problem with my SCSI bus. It was much slower than I thought it should be. When I used a new SCSI cable that was six feet longer than the one it replaced, the system performance returned to what I think it should be. Why?
- Q. 38. How can I connect a HVD device to my single-ended bus? [or vice versa]
- Q. 39. How can I connect a HVD device to my LVD bus?
- Q. 40. I have a LVD/MSE bus with several multimode peripherals, and I want to add a single-ended peripheral to it. I know that LVD/MSE is backward compatible through the single-ended interface, but when I add the single-ended peripheral, the LVD peripherals really slow down. Help!
- Q. 41. How can I connect a LVD peripheral to my single-ended SCSI bus?
- Q. 42. Why does my LVD/MSE bus slow down when I connect even one single-ended peripheral?
- Q. 43. I connected what I believed to be a single-ended peripheral to my multimode (LVD/MSE) bus, thinking it would switch to the single-ended mode and operate, although at a slower speed. The bus simply shut down. What happened?
- Q. 44. How can I connect a LVD peripheral to an HVD SCSI bus?

SCSI FAQ from Paralan

- Q. 45. What are LUNs?
- Q. 46. Can I have more than one computer on my SCSI bus?
- Q. 47. What is the difference between SCSI and IDE (or EIDE or ATAPI)?
- Q. 48. Is SCSI or IDE better for me?
- Q. 49. Will my single-ended to differential converter also do differential to single-ended conversion?
- Q. 50. What is termination?
- Q. 51. SCSI requires terminators. Where should they be placed?
- Q. 52. What is the difference between passive and active terminators?
- Q. 53. Should I use passive or active terminators?

Parts 3 and 4 Questions

Part 1 Q. | Part 1 A. | Part 2 Q.| Part 2 A. Part 3 Q. | Part 3 A. | Part 4 Q.| Part 4 A.

About Paralan | Products | Literature | About SCSI | Support | News | How to Buy

Paralan Corporation 4655 Ruffner Street, San Diego, CA 92111
Tel.: (858) 560-7266 | Fax: (858) 560-8929 | E-mail: **info@paralan.com**

SCSI Glossary

FAQ Answers

Part 1 Q. | Part 1 A. | Part 2 Q.| Part 2 A. | Part 3 Q. | Part 3 A. | Part 4 Q.| Part 4 A.

SCSI FAQ ANSWERS PART 1

Q. 1. What does the term "SCSI" mean?

- **Answer:** The term "SCSI" is an acronym for Small Computer System Interface. In the 1970s the name was appropriate. Today, SCSI is used for PCs, workstations, servers, mainframes, supercomputers.

Q. 2. What is SCSI?

- **Answer:** The Small Computer System Interface is a high-speed, intelligent peripheral I/O bus with a device independent protocol. It allows different peripheral devices and hosts to be interconnected on the same bus. Depending on the type of SCSI, you may have up to 8 or 16 devices connected to the SCSI bus. The number of devices can be dramatically expanded by the use of LUNs (Logic Unit Numbers). There must be at least one initiator (usually a host) and one target (a peripheral device) on a bus. There is a large variety of peripheral devices available for SCSI, including hard disk drives, floppy drives, CDs, optical storage devices, tape drives, printers and scanners to name a few. There are many implementations of SCSI starting with SCSI-1 to SCSI-2 to SCSI-3 including, Narrow, Wide, Fast, Ultra, Ultra-2 and Ultra160 SCSI. The SCSI specifications are approved and issued by ANSI and are developed by the X3T10 SCSI Committee.

Q. 3. What can I do with SCSI?

- **Answer:** SCSI provides a high-speed, intelligent interface that allows an easy connection for up to 16 devices (8 devices for Narrow SCSI) on a single bus. These devices may be hard disks, floppy disks, CDs, tape drives, printers and scanners to name a few. Peripherals may be mounted in the computer or in an external enclosure. Total SCSI cable length is dependent on the type of SCSI.

SCSI FAQ from Paralan

Q. 4. I seem to remember hearing the term SASI in the past. What is it?

- **Answer:** SASI is the acronym for Shugart Associates System Interface. It was developed in the 1970s by Shugart, at the time a dominant manufacturer of disk drives. It was meant to be an intelligent interface for disk drives only. Offering only 8-bit (Narrow), single-ended, asynchronous operation, by today's standards it was very slow (1.5 Mbytes per second). The standard connector for in-cabinet cabling is the non-shielded, 50-pin, female, low-density, connector having two rows of 25 pins each on 0.1 inch spacing. The standard connector for cabling outside the cabinet is the shielded, 50-pin, male, "centronics" type connector. In 1981 Shugart and NCR submitted SASI to the ANSI committee X3T9.2 as an open architecture I/O bus for disk drives. ANSI accepted the project, changed the name to Small Computer System Interface and added some major improvements to the specification. It was approved in 1986 by ANSI as document IEEE X3.131-1986. Today it is called SCSI-1. SASI is now long obsolete and, although many aspects of SCSI were backward compatible with SASI, it is very problematic.

Q. 5. Does SCSI work in both directions?

- **Answer:** Yes. SCSI is a bi-directional bus and will not work at all if it does not work in both directions. That also means that SCSI expanders such as a single-ended (SE) to differential converter will work as a SE to differential or a differential to SE converter. In other words, it does not make any difference if the initiator is on the SE side or on the differential side of the expander.

Q. 6. What are the differences between SCSI-1 and SCSI-2?

- **Answer:** The initial implementation of SCSI (now called SCSI-1) was designed primarily for Narrow (8-bit), single-ended, synchronous or asynchronous disk drives and was very limited relative to today's SCSI. It includes synchronous and asynchronous data transfers at speeds up to 5 Mbytes/sec. Only passive termination was defined. It did not include definitions of a device independent interface. The standard connectors are the familiar 50-pin, female, low-density (0.1 inch spacing), non-shielded connector (now termed the non-shielded Alternative 2, A-connector) for internal wiring and the equally familiar 50-pin, male, shielded "centronics" type connector for external wiring (now termed the shielded, Alternative 2, A-connector). This "centronics" type connector is frequently called the "SCSI-1 connector". 5 Mbyte/sec SCSI is termed "Slow" SCSI. SCSI cable lengths may be up to 6 meters (20 ft) for Slow SCSI. Even before X3.131-1986 was officially accepted by ANSI, the SCSI committee went to work on improving it.

  Released by the ANSI Committee as specification IEEE X3.131-1994, SCSI-2 is also a complete, stand-alone document. Arguably the most significant addition of SCSI-2 is the expanded definition of the common command set (CCS) providing a common software interface for all disk drives and many peripherals other than disk drives. SCSI-2 defines the differential interface and the 16-bit and 32-bit "Wide" data bus; doubles data throughput to 10 Megatransfers per second (called "Fast" SCSI), which translates to 10 Mbytes/sec for Narrow (8-bit) SCSI and 20 Mbytes/sec for Wide (16-bit) SCSI; adds the smaller 50-pin, high density, micro-D connector (termed Alternative 1, A-connector); and terms all 50-pin cables "A" cables. This 50-pin high-density connector is commonly called the "SCSI-2 connector". SCSI-2 recommends active terminators in place of passive terminators for the single-ended bus. Backward compatible to SCSI-1. Note that in SCSI-2 the 16-bit bus requires two cables (one "A" cable and one "B" cable) to make a connection. This seriously limited growth of the Wide bus. SCSI-2 maximum recommended single-ended SCSI cable length is up to 3 m (10 ft) for Fast SCSI. Differential cable length is 25 m (82 ft) for Fast or Slow SCSI.

Q. 7. What are the differences between SCSI-2 and SCSI-3?

- **Answer:** SCSI-3 changes the complete SCSI document structure and is no longer one document but a collection of documents, each with its own revision number. Some of these documents are the SCSI Primary Command (SPC) set layer, SCSI Block Commands (SBC) for hard disk interface, SCSI Stream Commands (SSC) for tape drives, SCSI Controller Commands (SCC) for RAID arrays, Multimedia Commands (MMC) , Media Changer Commands (MCC) and the SCSI Enclosure Services (SES) commands. For a complete overview see the **SCSI Architecture Model (SAM)** on the **T10 Committee Website**.

Let's take a look at some other important SCSI-3 documents:

- SPI
  The SCSI Parallel Interface (SPI) defines the electrical signals and connections for parallel SCSI. A very quickly adapted new feature defined in SCSI-3 is the 68-pin, high density, micro-D connector for 16-bit Wide SCSI (termed the Alternative 3, P-connector). The SCSI specification terms cables with this connector the "P" cable. This connector eliminates the necessity of using two cables for 16-bit SCSI and gave a tremendous boost to the growth of Wide SCSI. It is commonly referred to as the "SCSI-3" connector.

  There are several revisions of the SPI document. SPI includes Fast SCSI data transfer speeds up to 10 Megatransfers (20 Mbytes/sec for 16-bit). The Ultra SCSI (Fast-20) modification of SPI includes doubling the data throughput to 20 Megatransfers/sec (40 Mbytes/sec for 16-bit). Ultra SCSI speeds reduce the maximum single-ended cable length to 1.5 m (5 ft) with 5 or more devices and 3 m (10 ft) for systems having up to 4 devices. The maximum recommended differential cable length remains at 25 m (82 ft).
- SPI-2
  SPI-2 doubles bus speed again to the Ultra 2 (Fast-40) SCSI data throughput of 40 Megatransfers/s (80 Mbytes/s for 16-bit). To attain this speed, a new electrical interface is defined. This interface uses 3 V logic instead of TTL voltage levels and is known as Low Voltage Differential (LVD) SCSI. The older TTL based differential SCSI is now called High Voltage Differential (HVD) and it is not compatible with LVD signals. Most LVD device interfaces are designed as LVD/SE.

  Multimode operates at the LVD voltage levels and bus speed as long as all devices connected are LVD. Connecting a single-ended device to a multimode LVD bus causes all LVD/SE devices to switch to the single-ended interface. It will then operate at a maximum of 20 Megatransfers/sec (40 Mbytes/sec for 16-bit) with single-ended cable length limitations. Connecting an HVD device to an LVD bus will cause the bus to shut down. LVD cable length is specified as 12 m (40 ft). For a single initiator-single target application this length may be increased to as much as 25 m (82 ft). Note that single-ended signals cannot be used for bus speeds greater than Ultra SCSI (Fast-20).

  The low power requirements of the LVD interface allow the differential drivers to be included on the interface ASIC. Not having to place external driver chips on the PCB reduces the amount of PCB real estate required and reduces the cost of the board design.

Another new feature of SPI-2 is the SCSI Interlock Protocol (SIP) which defines the parallel command set. Also, SPI-2 adds two new SCSI connectors:

1. The 80-pin Single Connector Attachment (SCA-2) connector (termed the non-shielded Alternative 4, P-connector) that includes the 16-bit SCSI signals as well as power for the peripheral. This connector is designed for hot swapping of peripherals in SCSI backplanes.
2. The Very High Density Cable Interconnect (VHDCI) connector (termed the shielded Alternative 4, P-connector) is a small connector that allows as many as four separate 68-pin Wide SCSI connectors to be placed on one standard width PC backplate. Some of the newer LVD host adapters include this connector.

SPI-2 is a complete stand-alone document for all parallel interfaces up to Ultra 2 (Fast-40) SCSI and does not refer to older documents. To do this, it has incorporated the 50-conductor "A" cables defined in SCSI-2 and the 68-conductor "P" cables defined in the original SPI document.

- SPI-3
  SPI-3 again doubles the SCSI bus speed to Ultra 3 (also known as Ultra160 and Fast-80) providing SCSI bus speeds up to 80 Megatransfers/sec (160 Mbytes/sec for 16-bit). For this speed, clocking on both the rising and falling edges of the REQ and ACK clock is required. This is called Double Transition (DT) clocking and is defined for the 16-bit bus only.

  SPI-3 also includes a 32-bit CRC (Cyclic Redundancy Check) for better data security and Domain Validation. Domain Validation is new for peripheral buses. Basically, SCSI Domain Validation will not accept a negotiated data throughput speed until a validation test is performed. To perform this test, the initiator sends out a Write Buffer command to the target at the full data throughput. The initiator will then read the data back to see that it is correct. If it is not, the initiator will switch to the next lower speed and perform the test again. When the test passes, that speed is compatible with both the initiator and the target and is used for data transfers between the two devices.

  SPI-3 is also a complete document defining parallel SCSI interfaces up to 80 Megatransfers/sec and does not refer to previous SCSI documents. SPI-3 obsoletes HVD and 32-bit data bus designs. For specifications of the HVD and 32-bit bus, refer to SPI-2. The maximum cable length for Ultra 3 SCSI is 12 m (40 ft) or 25 meters (82 ft) for point-to-point applications.

  Ultra 160 (U160/m) is a sub-set of Fast-80 that includes Double Transition clocking, CRC and parts of Domain Validation. It is not yet a recognized form of SCSI.

  EPI
  For Paralan a very significant development released in the Enhanced Parallel Interface (EPI) is the documentation of SCSI Expanders, Bridging Expanders, Switches and some connectors not otherwise documented. This finally incorporates into the SCSI specification the types of products that Paralan has been designing, marketing and selling for years. EPI also describes the design of SCSI systems, defining the electrical specifications for cable lengths and loads. Also included is a description of how to work with both Wide (16-bit) and Narrow (8-bit) devices on the same SCSI bus.

SCSI FAQ from Paralan

Q. 8. What is the difference between single-ended and differential SCSI?

- **Answer:** Single-ended and differential are two methods of placing SCSI signals on the cabling. Single-ended uses one wire driven against ground and the signal is the voltage difference between that wire and ground. The differential interface drives two wires. The signal is the voltage difference between the two wires. Single-ended and differential are not directly compatible. (It should be noted that HVD and LVD are also not directly compatible). They can be interconnected by the use of a SCSI expander called a Single-ended to Differential Converter. Single-ended cable lengths are 6 to 1.5 meters (20 to 5 ft), decreasing with increasing data throughput, while differential (HVD and LVD) offers cable lengths to 25 meters (82 ft), regardless of the speed of the bus.

Q. 9. What is meant by "Narrow" SCSI?

- **Answer:** Narrow SCSI is the term that is used for 8-bit SCSI. It can usually be identified by 50-pin connectors.

Q. 10. What is meant by "Wide" SCSI?

- **Answer:** Wide SCSI is the term that is used for 16-bit SCSI. It can usually be identified by 68-pin connectors. From SCSI-2 until the SPI-3 document in SCSI-3, this term also applied to 32-bit SCSI. SPI-3 obsoleted the 32-bit SCSI bus.

Q. 11. What is HVD SCSI?

- **Answer;** This is the "old" differential SCSI using TTL voltage levels that was originally defined in SCSI-2, offering 25 meter (82 ft) cable length. It was functionally replaced by LVD (Low Voltage Differential) SCSI in the SPI-2 document of SCSI-3 and obsoleted in the SPI-3 document of SCSI-3. HVD and LVD SCSI are not directly compatible but can be interconnected by the use of a SCSI expander called an LVD to HVD Converter.

Q. 12. What is Wide Ultra SCSI?

Answer: Ultra SCSI, defined in the SPI-2 document of SCSI-3 offers a maximum data throughput of 20 Mbytes/sec for Narrow (8-bit) SCSI. Ultra Wide SCSI is the 16-bit version that offers 40 Mbytes/sec data transfers. Ultra Wide single-ended SCSI has a maximum cable length of 1.5 m (5 ft) with more than 4 active IDs and 3 m (10 ft) with 4 or fewer active IDs. Ultra Wide differential SCSI has a maximum cable length of 25 m (82 ft).

Q. 13. What is LVD SCSI?

Answer: LVD, which stands for Low Voltage Differential, was introduced in the SPI-2 document of SCSI-3. It is also called Ultra 2 or Fast-40 SCSI. It uses 3 volt instead of 5 volt logic level and is not directly compatible with the "old" differential (HVD) SCSI. LVD again doubles SCSI data throughput to 40 Megatransfers/sec. Cable lengths are 12 m (40 ft). Single initiator-single target applications may use up to 25 m (82 ft) of cable. The "multimode" implementation of LVD is backward compatible with single-ended SCSI. However, connecting one single-ended peripheral to a multimode LVD bus will cause the entire bus to switch to the single-ended mode with the single-ended limitations on data throughput and cable length. LVD can be interconnected with HVD by the use of a SCSI expander called an LVD to HVD Converter.

Q. 14. What is "multimode LVD" or LVD/MSE SCSI?

Answer: Multimode LVD and LVD/MSE (Multimode Single-Ended) are terms for the same interface. It is an implementation of SCSI that automatically switches between the LVD and the single-ended mode. When a single-ended device is connected to a multimode LVD/MSE bus, the entire bus switches to the single-ended mode. Otherwise LVD/MSE devices operate in the LVD mode.

Q. 15. What are the benefits of LVD SCSI?

Answer: In addition to the obvious benefits of longer maximum cable length than single-ended and a doubling of data throughput, there are a number of other benefits. LVD/MSE and single-ended offer some compatibility. The lower operating voltage of the LVD bus means lower power dissipation, so the differential drivers can be included on the LVD ASIC rather than having to mount them external to the chip. This results in smaller boards, less heat dissipation, higher reliability and lower cost. Also, manufacturers will no longer have to design and build devices with both single-ended and differential interfaces. This results in lower costs.

Q. 16. Is LVD SCSI backward compatible?

Answer: LVD is backward compatible through the single-ended interface if it is *multimode* LVD. It is doubtful that anyone will build LVD devices that are not multimode. Remember that connecting a single-ended device to a LVD/MSE bus will cause the entire bus to switch to the single-ended mode with its data throughput and cable length limitations. To add a single-ended peripheral to an LVD bus and preserve the data throughput and cable length of LVD, you can use a SCSI expander called an LVD to SE or LVD/MSE to LVD/MSE converter. This converter divides the SCSI domain into two bus segments - one segment will operate at the LVD data throughput and cable length and the other bus segment will operate at the single-ended data throughput and cable length.

Q. 17. What is Ultra160 or U160 SCSI?

Answer: Ultra 160 is defined in SPI-3. It offers data throughput of 80 Megatransfers/sec or 160 Mbytes/sec for Wide (16-bit) SCSI which is the only defined bus width. For this speed, clocking on both the rising and falling edges of the REQ and ACK clock is required. This is called Double Transition (DT) clocking. Also called Fast-80 or Ultra 3 SCSI.

Q. 18. I have heard of U160/m SCSI. What is it?

Answer: The SPI-3 document defines 5 new features for SCSI: Double Transition Clocking, CRC, Domain Validation, Quick Arbitration and Select (QAS), and Information Units (Packetization). In order to be compliant with the SPI-3 U160 specification, at least one of these features must be implemented. A group of industry leaders agreed to incorporate three of these features in order to speed up introduction of U160 products. These three features are Double Transition Clocking, CRC and Domain Validation. U160 devices with these three features are called U160/m.

Q. 19. Is Ultra160 SCSI backward compatible?

Answer: Ultra 160, also called Ultra 3 is backward compatible through the single-ended interface, if it is multimode Ultra 160. It is doubtful that anyone will build Ultra 160 devices that are not multimode. Remember that if a single-ended device is placed directly on a multimode Ultra 160 bus the entire bus will switch to the single-ended mode with its limitations on data throughput and cable length.

Q. 20. Is Ultra 160 SCSI better than fibre channel?

- **Answer:** This is a discussion that will go on for some time and there is no simple answer. It depends on the application. At least until now, fibre channel (FC) implementations of SCSI have been FC-AL or FC-Arbitrated Loop, so I will limit my comments to FC-AL. At the time of writing this FAQ, SCSI is beginning to ship devices that are capable of 160 Mbytes/sec data transfer rate while fibre channel (FC) is stalled at a maximum of 100 Mbytes/sec and the only peripherals with true FC interface are disk drives -- and only one manufacturer makes them. Mark up the data throughput advantage to SCSI.

  FC proponents say that connectivity is more important than data throughput and that FC can have up to 126 nodes. If that were true why would every increase in SCSI data throughput be immediately adopted? Data throughput rules in nearly every serious application we have encountered. Anyway, on a practical basis, FC is limited to only a couple dozen nodes which is very similar to what SCSI can handle. Call the connectivity issue a draw.

  Well then, FC can have up to 10 kilometers of fiber. Well, there are a number of manufacturers of SCSI Extenders that offer the same or longer fiber cable lengths. Call this one a draw.

  Again, at the time of this writing, an issue that FC proponents do not raise is interconnectivity. There are great problems in getting FC devices from different manufacturers to work together. Sometimes this is true with products from the same manufacturer. SCSI went through these problems many years ago. There are now very few problems with SCSI interconnectivity. Of course, these FC interconnectivity problems will eventually be solved, but for now, score this one for SCSI.

  Cost is always a factor. FC is more expensive. As FC gets wider application its cost will go down, but for now there is no contest. Score this one for SCSI.

  Overall, at this time, FC has a place in large scale storage and backup systems but there is almost no reason for the home user and very little reason for the small business to consider FC.

SCSI FAQ from Paralan

Q. 21. Is Ultra 160 SCSI better than EIDE?

- **Answer:** Again, the answer depends on the application. SCSI is an intelligent interface that can perform data transfers with no intervention from the host CPU. SCSI is multi-tasking. An initiator can issue a command to a target. The target can then disconnect from the bus to perform the task and free the bus up for another task. Ultra 160 SCSI can have up to 16 devices connected to the bus and they can be any of a large variety of peripherals, including hard drives, floppy drives, tapes, CDs, scanners, printers, etc. The number of devices can be substantially increased through the use of LUNs. EIDE can have two internal drives connected. Your PC probably has two EIDE buses, so it may have up to four peripherals. Ultra 160 SCSI allows up to 12 m (40 ft) of cabling, which may be internal or external to the computer. For point to point applications you may have up to 25 m (82 ft) of cable. EIDE is for internal cabling only and the maximum cable length is only 18 inches. And, don't forget that 160 Mbytes/sec is much faster than any EIDE bus.

  For a home user with a single hard drive, EIDE is probably better as it is less expensive and almost as fast as Ultra SCSI. In a compute-intensive or storage-intensive application, however, SCSI is the clear choice.

Q. 22. What is Fast-20 [or Fast-40 or Fast-80] SCSI?

- **Answer:** The term "Fast-xx" refers to the maximum data throughput that a particular version of SCSI is capable of, expressed in Megatransfers/sec. For example, Fast-20 is 20 Megatransfers/sec which is 20 Mbytes/sec for 8-bit (Narrow) SCSI and 40 Mbytes/sec for 16-bit (Wide) SCSI.

- Fast-10 is the same as Fast SCSI
- Fast-20 is the same as Ultra SCSI
- Fast-40 is the same as Ultra 2 (uses LVD transmissions)
- Fast-80 is the same as Ultra 3 or Ultra 160 SCSI (uses LVD transmissions)

Q. 23. Can I connect an Ultra 2 Wide (LVD) disk to an Ultra Wide adapter?

- **Answer:** The answer is definitely yes. However, exactly how you do it depends on the type of Ultra Wide host adapter you have. If it is single-ended SCSI and the LVD disk is multimode LVD, you can connect the disk directly to the host adapter. The multimode LVD interface on the disk will switch to the single-ended mode.

  If the Ultra Wide adapter is differential (HVD), they can still be connected, however, you will have to use a SCSI expander called an HVD to SE or an HVD to LVD Converter.

Q. 24. What is Double Transition clocking?

- **Answer:** Double Transition (DT) clocking is used to double the data transfer rate from Ultra 2 (Fast-40) to Ultra 3 (Ultra

160 or Fast-80) SCSI without having to increase the clock speed. That means that both edges of the REQ and ACK signals are used to clock data. The REQ and ACK signals run at 40 MHz on Ultra SCSI, so double clocking increases the rate at which data is clocked to 80 MHz. This provides data throughput of 160 Mbytes/sec for Wide SCSI.

Q. 25. What is "Domain Validation"?

Answer: Domain Validation is a method used in Ultra 160 (Fast-80 or Ultra 3) to test for the optimum rate for data exchange. Once the host adapter (initiator) has located a peripheral and negotiated a data transfer rate, the initiator sends a Write Buffer command to the target at that negotiated data transfer rate and then reads it back to determine if what it reads is what it wrote. If not, it will resend the Write Buffer command at the next lower data transfer rate. This will continue until a speed is reached where the test is successful. This is all accomplished automatically.

Q. 26. What is CRC?

Answer: Cyclic Redundancy Check is a means of detecting errors that is much more effective than the simple parity check that SCSI has used for years. CRC detects all single bit errors, all two bit errors, all errors with an odd number of bit errors, and all burst errors up to 32-bits long. CRC uses a 32-bit polynomial checksum to test data integrity. A similar process is used in Fibre Channel, Ethernet and other buses.

Part 1 Q. | **Part 1 A.** | **Part 2 Q.**| **Part 2 A.** | **Part 3 Q.** | **Part 3 A.** | **Part 4 Q.**| **Part 4 A.**

FAQ Answers

About Paralan | **Products** | **Literature** | **About SCSI** | **Support** | **News** | **How to Buy**

Paralan Corporation 4655 Ruffner Street, San Diego, CA 92111
Tel.: (858) 560-7266 | Fax: (858) 560-8929 | E-mail: **info@paralan.com**

EXPLANATIONS
- Auto
- Computer
  - Hardware
  - Internet
  - Peripherals
  - Security
  - Software
- Electronics
- Entertainment
- Health
- Home
- Money
- People
- Science
- Travel

EXPERT REVIEWS
Consumer Guide Auto
Consumer Guide Products
Mobil Travel Guide

OPINIONS
Member Home
Log In/Register

PRICES
Shop HowStuffWorks

VIDEO CENTER
Search Video Center

REFERENCE LINKS

Main > Computer > Hardware
PRINT EMAIL

How SCSI Works

by Tracy V. Wilson and Jeff Tyson

Inside This Article

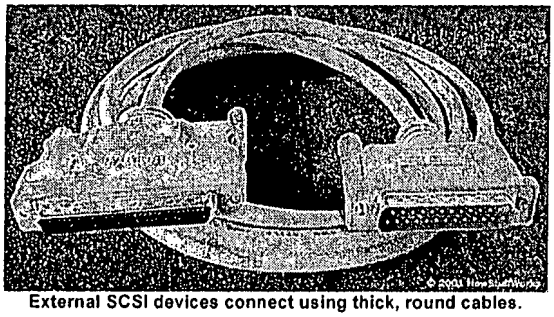
Controllers, Devices and Cables

A SCSI controller coordinates between all of the other devices on the SCSI bus and the computer. Also called a **host adapter**, the controller can be a card that you plug into an available slot or it can be built into the motherboard. The SCSI BIOS is also on the controller. This is a small ROM or Flash memory chip that contains the software needed to access and control the devices on the bus.

Each SCSI device must have a unique **identifier** (ID) in order for it to work properly. For example, if the bus can support sixteen devices, their IDs, specified through a hardware or software setting, range from zero to 15. The SCSI controller itself must use one of the IDs, typically the highest one, leaving room for 15 other devices on the bus.

Internal SCSI devices connect to a ribbon cable.

Internal devices connect to a SCSI controller with a ribbon cable. External SCSI devices attach to the controller in a **daisy chain** using a thick, round cable. (Serial Attached SCSI devices use SATA cables.) In a daisy chain, each device connects to the next one in line. For this reason, external SCSI devices typically have two SCSI connectors -- one to connect to the previous device in the chain, and the other to connect to the next device.

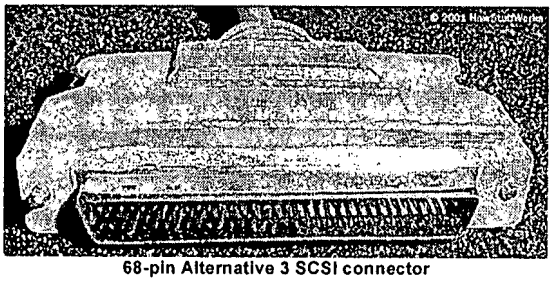


External SCSI devices connect using thick, round cables.

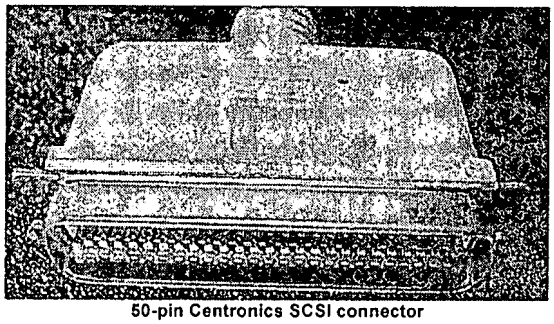
The cable itself typically consists of three layers:

- **Inner layer:** The most protected layer, this contains the actual data being sent.
- **Media layer:** Contains the wires that send control commands to the device.
- **Outer layer:** Includes wires that carry parity information, which ensures that the data is correct.

Different SCSI variations use different connectors, which are often incompatible with one another. These connectors usually use 50, 68 or 80 pins. SAS uses smaller, SATA-compatible connectors.



68-pin Alternative 3 SCSI connector



50-pin Centronics SCSI connector

Once all of the devices on the bus are installed and have their own IDs, each end of the bus must be closed. We'll look at how to do this next.

< PREVIOUS INTRO NEXT >

Inside This Article

SCSI Connectors and SCSI Cable information

SCSI Terms, SCSI specifications - SCSI cable length specs

The Connectors:

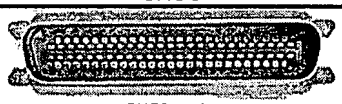
IDC50

IDC50 Female

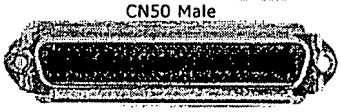
IDC50 Male

IDC 50 pin female, mates to IDC50 male "header", used on SCSI-1, SCSI-2, Ultra SCSI "narrow" etc. All internal 50-conductor "8-bit" SCSI uses these cconnectors.

CN50



CN50 Male



CN50 Female

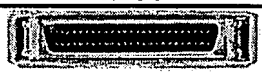
Centronics C50 sometimes referred to as CN50, Cen t50 (External connector on many SCSI-1 and some SCSI-2 Controllers) The controller connector and peripherals connectors should be females, and the cables male.

DB25

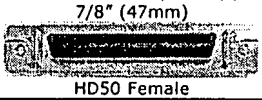
DB25 Male
SCSI-1
Used with the older Macs, Zip drives, and many scanners.

DB25 Female

HD50



HD50 Male
Mini D50 "HPDB50" (SCSI-2/SCSI-3 external connector) . The easy way to tell the difference between an HD50 and an HD68 (unless you want to count those tiny little pins) is to measure them. HPDB50 is about 1 3/8" (36mm) , and the HPDB68 is about 1 7/8" (47mm)

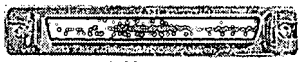


HD50 Female

HD68

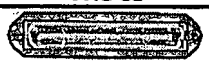
HD68 Male
MiniD68 "HPDB68" (Ultra wide SCSI-3/ Ultra2 LVD SCSI/ wide Differential SCSI both internal and external) The internal and external connectors appear to be different, but they

are the same basic connector. The easy way to tell the difference between an HD50 and an HD68 (unless you want to count those tiny little pins) is to measure them. HD50 is about 1 3/8" (36mm) , and the HD68 is about 1 7/8" (47mm).



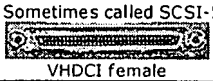
HD68 Female

VHDCI



VHDCI male

VHDCI 0.8mm HPCN68 male- Sometimes called SCSI-5. Very popular in RAID cards.



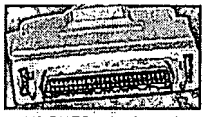
VHDCI female

HDI-30

HDI-30 is for Apple Macintosh PowerBooks.

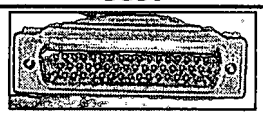
HPCN50

Used in Japan, on several Digital cameras and things.



HPCN50 pin (rare)

DB50



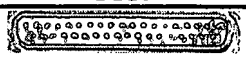
DB50 Male

SCSI-1

Usually used on old Sun Sparcstati ons.

DB50 Female

DB37

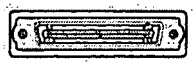


Male DB37

SCSI-1

Female DB37

HDCN60



Mini Centronics 60 "HDCN60" (old IBM RS6000)

We don't carry the se cables anymore

The SCSI Types Bus lengths, devices supported, etc.

Much more info on SCSI Cable lengths and "gotchas" below the chart

STA (SCSI Trade Association) -Endorsed Terms & Terminology for SCSI Parallel Interface Technology.

see here for term descriptions

STA Terms (notes-see below)	Bus Speed, MBytes/Sec. Max.	Bus Width, bits	Max. Bus Lengths, Meters (1)			Max. Device Support
			Single - ended	LVD	HVD	
SCSI-1(2)	5	8	6	(3)	25	8

Fast SCSI (2)	10	8	3	(3)	25	8
Fast Wide SCSI	20	16	3	(3)	25	16
Ultra SCSI (2)	20	8	1.5	(3)	25	8
Ultra SCSI (2)	20	8	3	-	-	4
Wide Ultra SCSI	40	16	-	(3)	25	16
Wide Ultra SCSI	40	16	1.5	-	-	8
Wide Ultra SCSI	40	16	3	-	-	4
Ultra2 SCSI (2,4)	40	8	(4)	12	25	8
Wide Ultra2 SCSI (4)	80	16	(4)	12	25	16
Ultra3 SCSI or Ultra160 SCSI (6)	160	16	(4)	12	(5)	16
Ultra320 SCSI (6)	320	16	(4)	12	(5)	16

Notes:
(1) The listed maximum bus lengths may be exceeded in Point-to-Point and engineered applications.
(2) Use of the word "Narrow", preceding SCSI, Ultra SCSI, or Ultra2 SCSI is optional.
(3) LVD was not defined in the original SCSI standards for this speed. If all devices on the bus support LVD, then 12-meters operation is possible at this speed. However, if any device on the bus is singled-ended only, then the entire bus switches to single-ended mode and the distances in the single-ended column apply.
(4) Single-ended is not defined for speeds beyond Ultra.
(5) HVD (Differential) is not defined for speeds beyond Ultra2.
(6) After Ultra2 all new speeds are wide only.

The Cable Length Rules*

(In case you're not confused yet)

A short simplified guide to scsi cable lengths.

Type of SCSI	"Single-ended" (Regular) SCSI bus length	"Differential" SCSI bus length	LVD SCSI bus length †† (ULTRA2 OR ULTRA160)
5 MHz (SCSI-1)	6 meters	25 meters	-
10 MHz (SCSI-2 FAST, Fast / Wide SCSI)	3 meters	25 meters	-
20 MHz (Ultra SCSI, Ultra Wide SCSI or "Fast20")	3 meters (3 devices + host adapter) or 1.5 meters (4 devices + host adapter)*	25 meters	-
40 MHz (Ultra2 SCSI or "Fast40")	-	-	12 meters

*please note: "Ultra" SCSI cable lengths are severely limited! The maximum cable length is ten feet when four devices (*including* the host adapter) or less are on the bus. If five devices are used (four devices and your host adapter), then the maximum bus length is 1.5 meters (five feet!). ††Knocking Ultra2 or U160 chains out of LVD mode by putting "Legacy" Single Ended (regular Ultra scsi, etc) devices on the chain will give you the same cable length restrictions as Ultra scsi. Watch it!!

Note: Remember, the "bus" is the entire cable chain! This is not some kind of "each cable" can be this length kind of deal. Bear in mind that you need to use good quality cable and active termination to acheive even these results! There are some specs that are mostly related to internal cables, for example, the SCSI-2 specs state that there should be 12" of cable between connectors. This spacing however, is not always possible, depending on the number of connectors. Another limit to be aware of is the "stub length" (sort of like the length from the cable "bus" to the device) is limited to 4", so don't chain adapters together if you can help it and never think you can do some kind of "Y" cable setup! Another possible enhancement involves using different spacing between the connectors to limit reflective resonance. There's some debate about whether this does, in fact, have any benefits. You can get extremely carried away with making "the perfect" cable, but you will generally pay far more than you get back. So, basically, remember that excess cable length is a bad thing, and if it works, and works reliably, it's just

fine, even if it bends a rule or two.

SCSI Products:

- **SCSI Adapters** (connector adapters)
- **SCSI Terminators** (Both SE and LVD)
- **Differential Converters** (HVD to LVD/SE)
- **SCSI External Cables** (outside computer)
- **SCSI Internal Cables** (inside computer)
- **SCSI Cards** (controller cards)
- **SCSI Ultra2/Ultra160 LVD cables**
- **SCSI VHDCI Cables**
- **SCSI Gender Changers** (M-M, F-F)

SCSI Information:

SCSI connector pictures
The basics of SCSI for newbies and technical information
SCSI connection FAQ
SCSI cable Length guide

Need more info? Check out the T10 Home page, brought to you by LSI Logic Corp.
Also excellent is the SCSI Trade Associations home page.
There are even more links available from our help page.

Not found what you need yet? Try these links:
Howto/Support - Contact us - OEM - Resellers

Site HDMI/DVI Music Stuff Audio/Video USB ieee-1394 SCSI Network I/O Cards Power Cooling

Toll Free: 888-726-2440
RAM Electronics Industries Inc.
7980 National Hwy. Pennsauken, NJ 08110

SCSI ID numbers

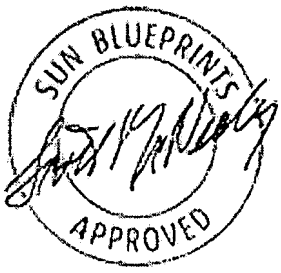
Part number 5500868,CDRSCS006AAWW

Each SCSI device attached to the host adapter, including the host adapter, must have a unique SCSI target address or SCSI ID. The address IDs are 0-7 for standard 8-bit SCSI devices and 0-15 for 16-bit Wide SCSI devices. The SCSI ID serves two purposes. It uniquely identifies each SCSI device on the SCSI bus, and it determines the device's priority on the SCSI bus.

Note: For SCSI devices connected to the AHA-2940 Ultra Wide Controller, SCSI ID 7 has the highest priority. The priority of the remaining IDs, in descending order, is 6 to 0 and 15 to 8. The AHA-2940 Ultra Wide Controller should remain at SCSI ID 7. If it is necessary to change the ID of the host adapter, go into the SCSISelect Utility by press the CTRL+A keys.

The 2940 Ultra Wide host adapter supports the SCSI Configured AutoMatically (SCAM) protocol, which assigns SCSI IDs dynamically and resolves SCSI ID conflicts at startup. If the computer includes SCSI devices that support SCAM, do not manually assign SCSI IDs to these devices. If a device does not support SCAM, disable the SCAM setting in the SCSISelect Utility.

If you install more than one SCSI host adapter in the computer, each adapter forms a separate SCSI bus. SCSI IDs can be reused as long as the ID is assigned to a device on a separate SCSI bus, for examples, each SCSI bus can have a device with SCSI ID 0. The host adapter with the BIOS loaded has priority for its device ID #0 if it is the bootable media.



SCSI-Initiator ID

By David Deeths - Enterprise Engineering

Sun BluePrints™ OnLine - August 2000

`http://www.sun.com/blueprints`

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303 USA
650 960-1300 fax 650 969-9131

Part No.: 806-6199-10
Revision 01, August 2000

SCSI-Initiator ID

Overview

Setting up a cluster is a complex task involving detailed configuration changes. Many of the changes involve the operating system, the volume manager, and sometimes high availability data services. These types of configuration changes are routinely made by system administrators and are commonplace—however, when a clustered system employs a shared SCSI chain there are additional configuration requirements which are performed using OpenBoot PROM (OBP) commands.

Because OBP commands are unfamiliar to many system administrators, implementing changes can be complicated. This article provides all necessary information for using OBP commands to change a host's SCSI-initiator ID.

Changing the SCSI-initiator ID is necessary for configurations using shared SCSI devices to function correctly when connected to multiple hosts—each SCSI ID on the chain must be unique (including the SCSI-initiator ID). Although there are numerous ways of configuring shared SCSI chains, the procedure described in this article can eliminate many of the potential problems faced when using other methods.

SCSI issues in clusters

When connecting multiple SCSI devices in a chain, it is important to ensure the SCSI IDs do not conflict. Each SCSI device requires a unique ID because the ID identifies a physical address on the SCSI bus.

To prevent conflict problems, the IDs for SCSI devices needs to be set. However, the methods used will vary depending on whether the device is a disk or host. For example, with disks, the ID can be set on the hardware, (although, with a Sun

storage enclosure, the ID is generally hardwired to a drive position within the enclosure—this method ensures that each position has a unique SCSI ID associated with the position).

Setting the SCSI ID for a host that is a SCSI-initiator is more complex.

The SCSI protocol supports two classes of devices:

Initiators

An initiator is generally a host device that has the ability to initiate a SCSI operation. The default SCSI-initiator ID is 7.

Targets

A target is a device (generally a storage device, although it could be a printer, scanner, or any other device on the SCSI chain), that will respond to SCSI operations.

Because targets generally require an initiator to function, they should not use ID 7. Available IDs fall within the ranges of 0-6 and 8-15. Narrow SCSI targets have IDs available in the 0-6 range, while wide SCSI have IDs available in the range of 0-15. If only one system is connected to a SCSI chain, the SCSI-initiator ID (the SCSI ID of the system) will not require changing.

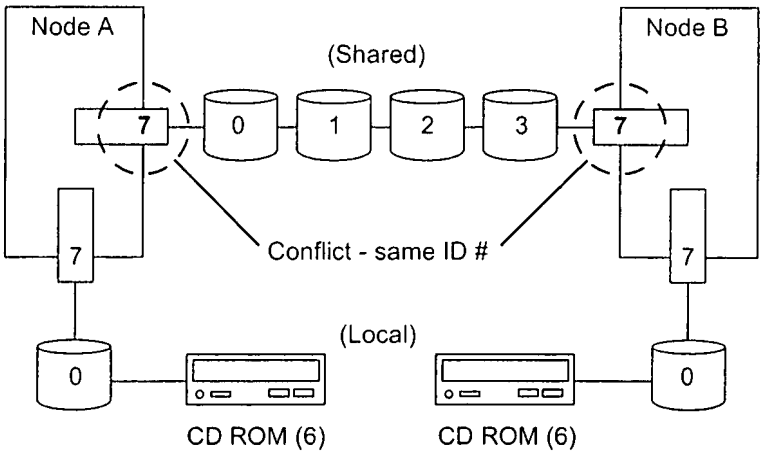
Each SCSI chain attached to a system is associated with a single SCSI controller. Each controller is the connection point of the system to a specific SCSI chain. The host can have a different address on each SCSI chain, therefore it can be useful to think of the SCSI address as belonging to the controller (the host's connection to a specific chain) itself, rather than to the host. The SCSI-initiator ID represents the system's address on a specific chain and can be set environment-wide for all controllers on the system, or individually for each SCSI controller. An environment-wide ID sets the default value for all controllers not having an ID explicitly set.

Clustered configurations are more complex to configure than a single machine because there could be multiple initiators on the SCSI chain. However, if only one initiator is on a chain, all targets will be local to the initiator—which means they can only receive requests from that initiator. In a single non clustered system configuration there can be several SCSI chains—each with its own set of SCSI IDs, however, each chain will still be local to the single initiator.

A clustered configuration can comprise a combination of local and global chains. The local chains will have connections from a specific host to target devices only, while the global chains will contain connections to target devices and to other initiators.

On local chains, each host must retain the default SCSI ID of 7 to prevent local chain conflicts—however, on global chains, the default ID on one of the initiators requires changing. If the ID is not changed, both initiators will share the same ID (and therefore the same address) as shown in FIGURE 1 which will result in a SCSI address conflict.

FIGURE 1 Conflicting SCSI-initiator IDs



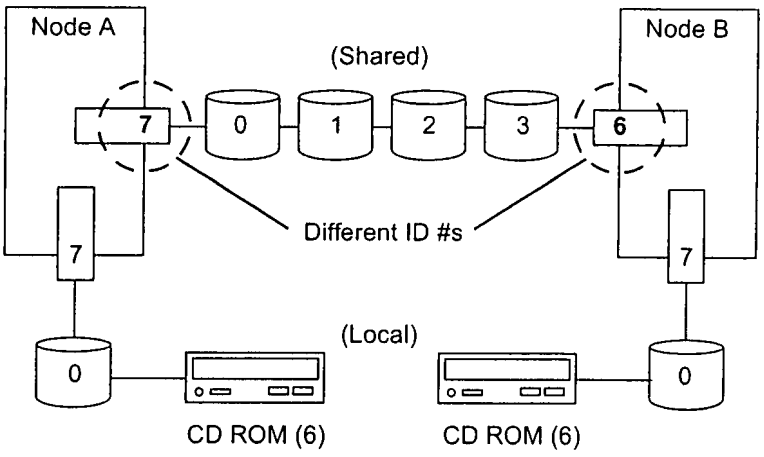
For local chains, the SCSI-initiator ID should be kept at the default value of 7—thereby preventing SCSI ID conflicts with devices such as the boot disk or CD-ROM (and any internal devices to be installed in the future).

For shared chains, one of the hosts should have its SCSI ID set to 6—this will prevent conflict problems (see FIGURE 2). The other host SCSI ID should be left at the default value.

Note – Setting the SCSI ID to 6 will prevent drives with a higher priority from hijacking the bus. Although changing the global SCSI-initiator ID of a device to 8 (or another unique ID on the chain) could solve SCSI initiator conflicts (and avoids the necessity of writing an nvram script), there could be a precedence problem because ID 8 has a precedence below that of other targets. A SCSI ID of 7 has the highest precedence, followed by 6, decreasing down to zero. IDs 8 to 15 have a precedence below ID zero.

When changing the SCSI-initiator ID, ensure none of the drives or other devices in the chain have the same ID. This holds true for global and local chains—on any given chain ensure each SCSI address is only used once. For example, in some Sun™ servers, the internal drives exist at IDs 0 and 1, and the CD-ROM has an ID of 6. Therefore, these IDs must be avoided by the SCSI-initiator ID.

FIGURE 2 Non-conflicting SCSI-initiator IDs



Changing SCSI initiator ID Overview

Any changes made to the SCSI-initiator ID should be made prior to connecting the host to the SCSI chain affected.

The procedure for changing the SCSI initiator ID requires modifying some OBP variables and setting up a script that will run as part of the system initialization. This can be performed using shell commands, or by working directly at the OBP level (the ok prompt). Changes made to an OBP environment variable are nonvolatile—all changes will be retained (even when the OS is reloaded).

The procedure outlined here should only be performed on one node of a 2 node cluster—a similar procedure could be followed for clusters with more than two nodes, however, the SCSI chains have to be mapped out, and a decision made as to which controllers on specific chains are to be changed. In most cases, it is best to have changes made on as few machines as possible. For example, in a 4-node Ring-Topology cluster, 2 nodes should remain unchanged, while the other two nodes should have the SCSI ID of the external controllers ID set to 6. The nodes with their IDs set to 7 would only share disks with the nodes that have their IDs set to 6. Remember, all IDs on a chain must be unique.

If you administer multiple clusters, it is a good idea to have a standard procedure for determining which device node will have the SCSI ID changed. For example, changing the initiator ID of the node with the highest ethernet address, or the node with the highest suffix on the host name (if the node names are the cluster name affixed with a number) will make it easier to remember which node has a non-default SCSI-initiator ID.

All changes made to the SCSI- initiator ID, and at the OBP level should be documented to assist with future troubleshooting.

Changing an environment-wide SCSI ID on one node of a two node cluster, and explicitly setting the local chains back to 7 eliminates conflicts on the shared chains and is transparent to devices on the local chains. A cluster environment usually has only one local SCSI chain, but could have several shared chains. This method reduces the number of controllers that need the SCSI IDs to be explicitly changed. Additionally, most cluster configurations will not require extra work to add a new shared chain—this is because any new controllers will have their SCSI IDs set to the default.

Note – It is important to ensure the host SCSI IDs do not interfere with each other, and also that target IDs do not interfere with the host IDs. Each SCSI ID on the chain must be unique—regardless of whether used by a target or initiator device—this could mean leaving an empty slot in a storage enclosure if the SCSI ID for the slot conflicts with the modified SCSI-initiator ID.

Note – Setting the SCSI ID for individual controllers is more complex than setting an environment-wide default. Both require using the OBP prompt or the Solaris Operating Environment `eeprom` command. Veritas Volume Manager (and other programs) store information in the NVRAM, therefore, it is important to check the NVRAM contents prior to making any changes. This is done using the `printenv nvramrc` OBP command, or the `eeprom nvramrc` command at the Solaris™ Operating Environment shell prompt. In most cases the existing NVRAM contents can be appended to the NVRAM script that sets the initiator ID. Some firmware upgrades will modify the OBP variables, see the section 'Maintaining Changes to NVRAM'.

The NVRAM script consists of OBP commands that are run in sequence. The script is called after system initialization (but before the device tree is built). Because a device tree must already exist for some OBP commands to work correctly, the NVRAM script must first run the probe-all command (which creates the OBP device tree). However because the probe-all command will also be run independently during the startup procedure, two OBP device trees would be created—only one of which will be seen by the operating system. The nvramrc changes will be effected in the tree which is not seen by the operating system. To prevent this happening, a procedure to keep the second `probe-all` operation from occurring needs to be invoked. The system start-up procedure has a work-around to prevent two sets of trees being created. This work-around will allow the running of the `probe-all` command in the NVRAM script, and prevents the `probe-all` command from running during a standard system start-up procedure (as listed in TABLE 1).

TABLE 1 System Start-up Process

Order	Procedure	Description
1	POST	Power on self-test
2	system initialization	
3	evaluate `nvramrc`	Evaluates the `nvramrc` script (if the `use-nvramrc?` OpenBoot environment variable is true.)
4	Run `probe-all` command*	Builds the device tree and initializes the devices
5	Run `install-console` command*	Sets up console (keyboard or terminal port)
6	Run `banner` command*	Prints system information to console
7	secondary diagnostics	
8	boot	Boots the system

* Only if the banner command is not called during the nvramrc script evaluation.

The work-around script uses the banner command (which displays system configuration information) in the NVRAM script to signal that the part of the standard system start up that runs the commands probe-all, install-console, and banner commands is being executed by the NVRAM script. If the banner command appears anywhere in the NVRAM script, the banner command will not run outside of the script (neither will the install-console or probe-all commands).

Note – Instead of using the banner command to signal the probe-all and install-console commands are to be run within the nvram script, the suppress banner command can be used (anywhere in the nvramrc script). The suppress-banner command does not display system information.

After the probe-all command has created the device node tree, it is then possible to change the device node properties (including the SCSI ID). The OpenBoot directory structure lists devices in a manner similar to the UNIX® device tree—each device node represents a level in the hardware hierarchy.

The peripheral bus (sbus or pci) is one of the main nodes under the root. All peripheral devices branch off this node—each device has a unique name which comprises several components:

- Device Identifier

The first part of the name identifies the device, however, the naming convention may differ for each type of device node. The device identifier is followed by the @ symbol.

Note – Device identifiers for peripheral cards are generally identified by the manufacturers symbol, followed by a comma, then the device type abbreviation (for example, SUNW,qfe@2,8c00000 identifies a quad-fast ethernet device manufactured by Sun Microsystems).

- Device Address and Offset

Following the @ symbol, the device address and offset identifier are listed—the address is listed first, followed by a comma, then the offset is listed, these identifiers generally reflect a connector or slot location. For example, the sbus@f,0 and sbus@e,0 entries in FIGURE 3 indicate two different sbus hardware slots.

The identifier-address-offset combination allows for multiple devices (of the same type) to be uniquely identified using the address and offset. Figures 3 and 4 show examples of OBP device trees. Some devices in the OBP device tree do not represent actual devices—these are termed pseudodevices. Other OBP devices represent hardware components unrelated to the peripheral cards, for example, in Figure 4, the CPU is represented by the name
"SUNW, UltraSPARC-II".

Although an understanding the OBP device tree is beneficial for comprehending the architecture of the machine, it is not mandatory for changing the SCSI-initiator ID.

FIGURE 3 Example of sbus OBP Device Tree (incomplete)

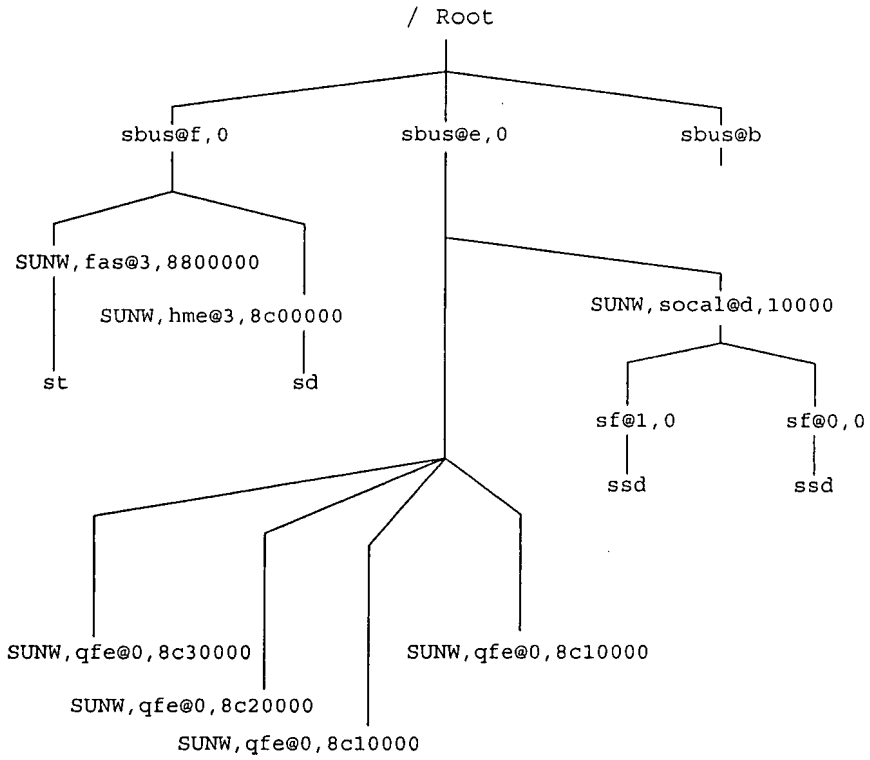
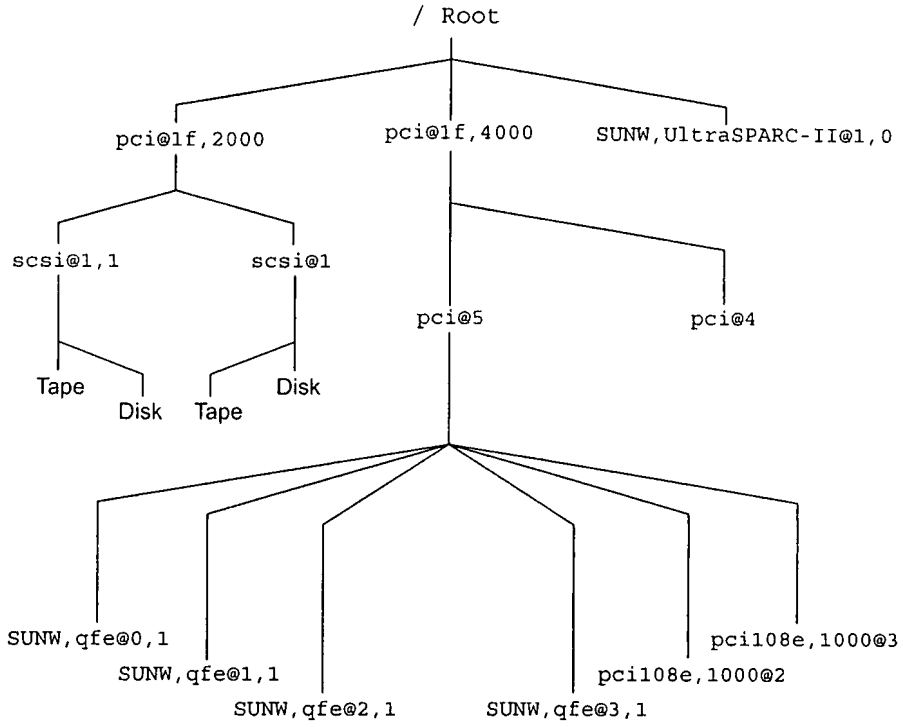


FIGURE 4 Example of pci OBP Device Tree (Sun Enterprise™ 250) (incomplete)



Mapping the OBP address to a physical card location can be difficult. To map the hardware location to an OBP node, use the following tables (2, 3, 4, and 5). These tables display the mapping between physical device slots and OBP device nodes for the current generation of Sun systems across several product families. Additional information can be found in the appropriate service manuals. The variable *devname* represents the name of the device in a given slot. Figure 5 displays the physical device slots referred to in TABLE 2.

FIGURE 5 I/O Board Interfaces in the E XX00 Series

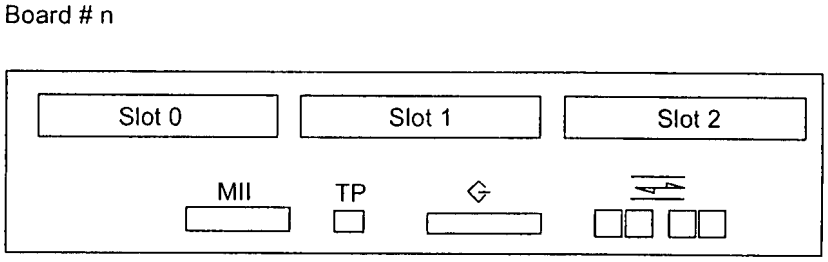


TABLE 2 E XX00 Series: Relationships of Physical Locations to OBP Nodes

Hardware Slot on Board # n	OBP Device Tree Node
0	/sbus@(2n+1)/*devname*@0
1	/sbus@(2n)/*devname*@1
2	/sbus@(2n)/*devname*@2
Ethernet (TP)	/sbus@(2n+1)/hme@3
SCSI	/sbus@(2n+1)/fas@3
Fiber	/sbus@(2n)/socal@d

TABLE 3 Sun Enterprise 450: Relationships of Physical Locations to OBP Nodes

PCI Slot Number (labeled)	OBP Device Tree Node
10	pci@1f,4000/*devname*@4
9	pci@4,4000/*devname*@2
8	pci@4,4000/*devname*@3
7	pci@4,4000/*devname*@4
6	pci@4,2000/*devname*@1
5	pci@1f,2000/*devname*@1
4	pci@6,2000/*devname*@1
3	pci@6,4000/*devname*@2
2	pci@6,4000/*devname*@3
1	pci@6,4000/*devname*@4

TABLE 4 Sun Enterprise 420R or Netra™ t1400/1405: Relationships of Physical Locations to OBP Nodes

PCI Slot Number (labeled)	OBP Device Tree Node
4	pci@1f,4000/*devname*@5
3	pci@1f,4000/*devname*@2
2	pci@1f,4000/*devname*@4
1	pci@1f,2000/*devname*@1

TABLE 5 Sun Enterprise 250, Sun Enterprise 220R, or Netra t1400/1405: Relationships of Physical locations to OBP Nodes

PCI Slot Number (labeled)	OBP Device Tree Node
3	pci@1f,2000/*devname*@1
2	pci@1f,4000/*devname*@2
1	pci@1f,4000/*devname*@4
0	pci@1f,4000/*devname*@5

Navigating through the device tree is accomplished in a similar manner as in the Solaris Operating Environment—by using the `cd` and `ls` commands. However, when working at the OBP prompt, you begin outside of the device tree. To gain access to the device tree, the `dev` command or an absolute path (one beginning with the root slash) must be used. To exit the device tree, use the `device-end` command.

The `.properties` command lists the properties of the current device node and can be useful for debugging problems when setting the SCSI-initiator ID. The following codebox is an example of an output from the `.properties` command run from a device node that represents a SCSI controller:

```
ok pwd

/sbus@1f,0/SUNW,fas@e,8800000

ok .properties

scsi-initiator-id        00000007

hm-rev                   00 00 00 22

device_type              scsi

clock-frequency          02625a00

intr                     00000020 00000000

interrupts               00000020

reg                      0000000e 08800000 00000010

                         0000000e 08810000 00000040

name                     SUNW,fas
```

In the above example, the current SCSI-initiator ID for this controller is listed in the `scsi-initiator-ID` field. Unless the value of the `scsi-initiator-id` field of the controller is set explicitly, it will default to the value of the `scsi-initiator-id` OBP environment variable.

The method for changing the ID is explained in the following sections.

Changing the SCSI-initiator ID

The NVRAM script can be edited from either a Solaris Operating Environment shell, or from the OBP prompt. The shell method is easier to use if the OS is operational. However, if the OS is down, it will be easier to use the OBP method. In either case, the machine will have to be rebooted for the changes to take effect.

This procedure is intended for a two node cluster. The issues involved in using a similar process for clusters with more than two nodes have been discussed earlier in the article.

Using the Shell Command Prompt Method

Set the environment-wide SCSI-initiator ID to 6 by using the `eeprom` command. This command must be run as the root user.

```
# eeprom scsi-initiator-id=6
```

Because the above change was made at the OBP level, the change becomes permanent (even if the OS is reloaded). To ensure the change does not interfere with local SCSI chains (including the CD-ROM and boot devices), you need to write a script that resets the local chain controller ID(s) back to 7.

The easiest way to create an NVRAM script at the shell prompt is to make a file that will contain a script of the OBP commands (*not* shell commands). The script is then stored in NVRAM using the `eeprom` command. Create this file using a text editor—store the file where it can be retrieved at a later date. The following is a step-by-step description of the commands needed in the script.

The first command required is the `probe-all` command. This command will be the first action taken by the `nvram` script—it will probe the devices and create the device tree. The file should now look as follows:

```
probe-all
```

Next, change the SCSI-initiator ID back to 7 on the local chains (which are not connected to other initiators). For each controller connected to a local chain, the script should enter the device node that represents the controller (using the `cd` command) and change the SCSI-initiator ID value. For example, if the internal / local controller is represented by the device node `SUNW,fas@e,8800000` connected to the sbus card at address `1f` (as with Ultra™ 1 and Ultra 2 systems), the following command would be added to the file:

```
cd /sbus@1f,0/SUNW,fas@e,8800000
```

Note – The device node path we use is only an example. In a situation where there are multiple local chains, the script will need to execute commands to enter each device node representing a local controller and explicitly change its initiator ID.

After entering the device node, change the SCSI-initiator ID to 7 by inserting the following line to the file:

```
7 " scsi-initiator-id" integer-property
```

Note – There should be a space before and after the first quotation mark. The quotation mark is a special OBP command that tells the tokenizer to treat the next word (which must be immediately followed by a quotation mark) as a string of characters. Therefore, the quote is separated from the string by a space because it is a command that operates on the string (not a syntactic marker as would be the case at the shell prompt).

The file should now look similar to the following:

```
probe-all
cd /sbus@1f,0/SUNW,fas@e,8800000
7 " scsi-initiator-id" integer-property
```

Change the SCSI-initiator ID for any other local chains to 7 (by using the method described previously).

After the SCSI-initiator IDs have been changed for all local chains, insert the `device-end` command into the file. Inserting this command will allow the script to exit the device tree. The file should now look similar to the following:

```
probe-all
cd /sbus@1f,0/SUNW,fas@e,8800000
7 " scsi-initiator-id" integer-property
device-end
```

As discussed in the beginning of this article, when the `probe-all` command is used in an NVRAM script, you also need to include the `banner` command in the file as a flag to keep the `probe-all` command from running again. Because the banner command also prevents the `install-console` command from running automatically, you must include `install-console` in the script as well. At the end of the file, include the `install-console` and banner commands. The file should now look similar to the following:

```
probe-all
cd /sbus@1f,0/SUNW,fas@e,8800000
7 " scsi-initiator-id" integer-property
device-end
install-console
banner
```

To store the file contents in NVRAM, run the `eeprom` command. The actual command to be run depends on the name of your file. For example, if your file is named `/nvram_file`, the command to use would be as follows:

```
# eeprom nvramrc="`cat /nvram_file`"
```

In the previous command, the *backtics* (`) will interpret the `cat` command to output the file. If the quotation marks are not included, only the first line will make it into the nvramrc variable. If the backtics are not included, the nvramrc variable will contain the unevaluated string `cat /nvarm_file`. Because different shells may treat backtics and quotes differently, check the variables when finished. The above syntax is suitable for the Bourne and Korn shells.

The `nvramrc` environment variable should now be correctly configured, however, the NVRAM script will not be run at initialization unless the `use-nvramrc?` environment variable is set to `true` using the `eeprom` command.

```
# eeprom use-nvramrc?=true
```

Confirm all set values by using the eeprom command, The eeprom command will display a value if you run it without specifying a new value with the equal sign. Enter the following commands to check the appropriate OBP variables:

```
# eeprom scsi-initiator-id
scsi-initiator-id=6
# eeprom nvramrc
nvramrc=probe-all
cd /sbus@1f,0/SUNW,fas@e,8800000
7 " scsi-initiator-id" integer-property
device-end
install-console
banner
# eeprom use-nvramrc?
use-nvramrc?=true
```

Using the OpenBoot PROM (OBP) Prompt Method

All steps performed in the previous section can also be implemented at the OBP prompt, however, the process differs slightly.

Note – The steps performed in this section are for an OBP prompt only and are not suitable for running at the shell command prompt.

Set the environment-wide SCSI-initiator ID to 6 by using the setenv command.

Enter the following command into the OBP prompt.

```
ok setenv scsi-initiator-id 6
```

Because the preceding command sets an OBP environment variable (stored in NVRAM), the change becomes permanent automatically. Therefore, a script should be created to reset the local chain controller ID(s) back to 7. This script can be created using the `nvedit` command—this command starts a basic editor that operates in insert mode only and uses keystrokes similar to EMACS (see Table 6). The `nvedit` editor displays a line number at the beginning of each line (which is the only way to know which line you are on).

The editor begins with line zero—the line you are currently editing is always displayed at the bottom of the screen. Pressing return at the end of any line will create a new line. If there are additional lines after a newly added line, they will have their line numbers adjusted to account for the new line.

Note – It is easy to unintentionally press return and move existing lines down without realizing it, therefore, it is a good idea to review the finished file using the Ctrl-p or Ctrl-n keystrokes.

To start the nvedit editor, enter the following command at the OBP prompt.

```
ok nvedit
   0:
```

Note – In the preceding codebox, the `'0'` is the first line number.

TABLE 6 The `nvedit` Commands

Keystroke	Action
Ctrl-c	Exits the `nvramrc` editor and returns to the OBP command interpreter. The temporary buffer is preserved, but is not written back to the `nvramrc` variable. (Use `nvstore` to write back.)
Ctrl-l	List all lines
Ctrl-p	Moves to previous line
Ctrl-n	Moves to next line
Ctrl-b	Moves back one character
Ctrl-f	Moves forward one character
Delete	Deletes a character
Enter/Return or Ctrl-o	Starts a new line
Ctrl-k	Delete (kill) all characters from the cursor to the end of the line (including the next carriage return). If used at the end of a line, it will join the current line with the next line. If used at the beginning of a line, it will delete the entire line.

On line zero, enter the `probe-all` command (followed by the return key). This will cause the NVRAM script to probe the devices and create the device tree. The NVRAM script should look as follows:

```
0:  probe-all
```

After pressing the return key, line zero will move up the page and the cursor will be on line 1.

For each controller connected to a local chain, use the cd command to enter the device node that represents the controller—reset the SCSI-initiator ID value back to 7. For example, if the internal / local controller is represented by the device node SUNW,fas@e (connected to the sbus card at address 1f),enter the device node by inserting the following command:

```
cd /sbus@1f,0/SUNW,fas@e,8800000
```

The SCSI-initiator ID value is changed by entering the following line:

```
7 " scsi-initiator-id" integer-property
```

Note – There should be a space before and after the first quotation mark.

In this example, the SCSI-initiator ID for the controller can be set in NVRAM when the previous two commands are added to lines 1 and 2 of the NVRAM script—as shown:

```
1: cd /sbus@1f,0/SUNW,fas@e,8800000
2: 7 " scsi-initiator-id" integer-property
```

The previous lines should be repeated for each local chain. After the IDs have been changed for all local devices, add the following commands to the end of the script (the line numbers used in the following codebox are example only):

```
3: device-end
4: install-console
5: banner
6:
```

The return keystroke following the banner command (line 5) will create a new line (6). Because the script is now completed, press Ctrl-c to exit the `nvedit` editor. Although the script is completed, it is not yet committed to NVRAM. To save the changes, use the `nvstore` command to move the `nvedit` buffer into NVRAM. Enter the following command into the OBP prompt:

```
ok nvstore
```

The nvramrc script should now be correctly configured, however, it will not be run unless the `use-nvramrc?` environment variable is set to `true` by using the `setenv` command. Enter the following command into the OBP prompt, note the output:

```
ok setenv use-nvramrc? true
use-nvramrc? =         true
```

Verify the contents of the `nvramrc` variable using the `printenv nvramrc` command. This command will display the contents of `nvramrc` variable. The display should look as follows:

```
ok printenv nvramrc
nvramrc =    probe-all
   cd /sbus@1f,0/SUNW,fas@e,8800000
   7 " scsi-initiator-id" integer-property
   device-end
   install-console
   banner
```

If there are any errors use the nvedit editor again to make the required changes. Move to the line(s) to be edited by using the Ctrl-p, or Ctrl-n keystrokes—edit the line, or delete it using the Ctrl-k keystroke.

After making changes, run the `nvstore` command again. When the `nvramrc` script is correct—restart the machine for changes to take effect. Enter the following command:

```
ok reset-all
```

Maintaining Changes to NVRAM

Another consideration is that some patches or system upgrades may alter the NVRAM, or disable the `use-nvramrc?` environment variable, and could even erase the NVRAM script. For example, the Solaris Operating Environment version 7 software includes a firmware patch to allow some older Ultra Enterprise™ 1 and Ultra Enterprise 2 systems to use 64 bit addressing. Although using this patch is optional, it will reset the `use-nvramrc?` environment variable and prevent the NVRAM script from running. After any changes are made involving firmware or EEPROM, check the values of any OBP variables that may have changed when using the `eeprom` command.

Run the following unix commands:

```
eeprom use-nvramrc?
eeprom nvramrc
eeprom scsi-initiator-id
```

Summary

Changing the SCSI-initiator ID on a cluster node allows two nodes to share SCSI storage devices on a single SCSI chain. This is essential for correct operation of a shared SCSI chain. Changing IDs requires either direct or indirect modification of the `nvramrc` script (which is a set of OpenBoot commands). Changes are permanent, even after reinstallation of the operating system.

Because some firmware upgrades can modify OBP variables, it is critical to check these after any firmware changes.

Author's Bio: David Deeths

David Deeths is a member of the technical staff of the Enterprise Engineering group, a part of Sun Microsystems' Computer Systems division. He has been working at Sun for 3 years. His current focus is clusters, including doing some of the research, development, and writing for the upcoming Sun Blueprints(tm) "Introduction to Clusters" book. David has degrees in Electrical Engineering and Cognitive Science from the University of California, San Diego.

Table of Contents

SCSI

CHAPTER 1

SCSI BUS OVERVIEW

1.1 Introduction

The Small Computer System Interface (SCSI) is a parallel I/O bus and protocol that permits the connection of a variety of peripherals including disk drives, tape drives, modems, printers, scanners, optical devices, test equipment, and medical devices to a host computer. This chapter provides an overview of the SCSI system.

1.2 The SCSI Bus

The SCSI bus connects all parts of a computer system so that they can communicate with each other. The bus frees the host processor from the responsibility of I/O internal tasks.

The SCSI protocol is a peer-to-peer relationship: one device does not have to be subordinated to another device in order to perform I/0 activities. A total of eight devices can be connected to the bus simultaneously. Only two of these devices can communicate on the bus at any given time.

NOTE:

The word "device" refers to the SCSI host adaptor or a device controller connected to the SCSI bus; it does not refer to the peripheral device.

A unique **SCSI device ID (7-0)** is assigned to the SCSI host adaptor and to each (device controller. One of the devices on the bus must be the initiator (usually the host adaptort). The other device is the target (usually the peripheral device controller). When two devices communicate on the bus, one device initiates the communication to the target, and the target performs the task. SCSI devices usually have a fixed role as an initiator or a target, although some devices can perform both roles.

The SCSI interface on Digital systems is a single-ended configuration with a cable length of up to six meters. The SCSI cable is always terminated on the SCSI host adaptor module; an external terminator must be provided for the last peripheral device placed on the bus. **Figure 1** shows a typical SCSI configuration.

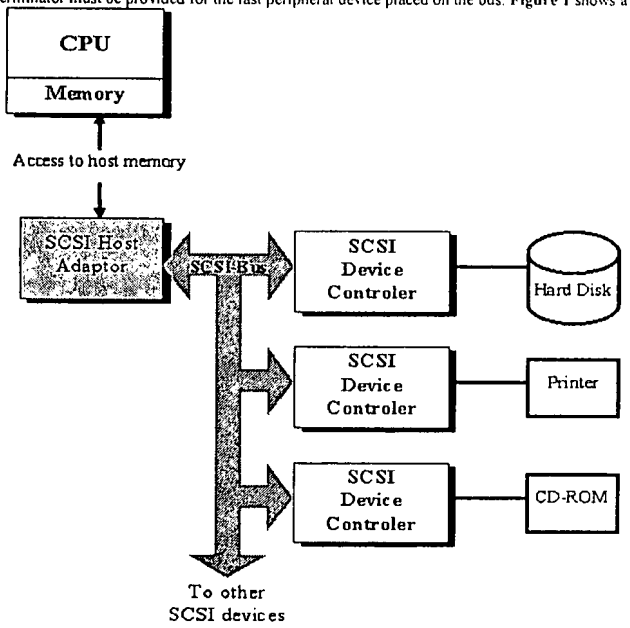
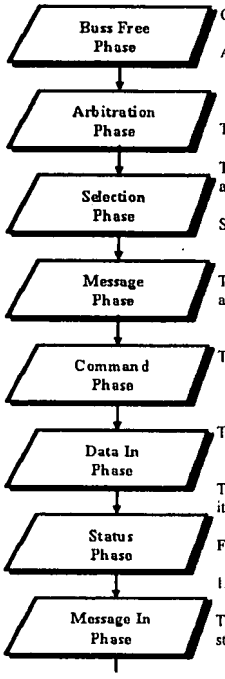


Fig. 1 SCSI Configuration

1.3 SCSI Bus Phases

The Small Computer System Interface bus can be time-shared, which results in greater usage of bus bandwidth. This is how it works: while one device is using the bus, other devices may be active and performing internal activities. Devices do not use the bus unless they are involved in data transfer or have status to report. Devices may disconnect from the bus while time-consuming activities internal to the device are occurring. As soon as a device is ready to resume communication, the device can arbitrate for the bus (when the bus is free) to reattach to the host. System performance is significantly increased when devices disconnect and reconnect to the bus. During the bus phases (Figure 2), devices must first contend for access to the bus. Then a physical path is established between the initiator and target. Remember, the SCSI bus cannot be in more than one phase at a time.



On a busy system, the SCSI bus may be free. for as little as 1.2ps while there is no device requosting the bus. Or, it could remain in the bus free state indefinitely.

A device can arbitrate and win the bus in 3.6ps or less. Devices that lose arbitration can try again when the bus is free.

The Selection phase can occur in 580 ns. If the target does not respond, the bus is free in about 250ms. The target controls the bus alter the selection phase.

This is tlie first information transfer phase in the connection. It allows the initiator to send an Identify message to the target. Messages are always transferred asynchronously.

Six Inquiry command bytes are transferred asynchronously from the initiator to the target.

The target responds with Inquiry data. The data is transferred synchronously it both the taraet and the initiator have previously established a synchronous data transfer agreement.

The target sends a single status byte asynchronously.

The last information that is transferred in the connection is typically the Command Complete message.

The target releases all signals within 10ls of the initiator's acknowiedgement of the previous message. The initiator releases any signals it may be driving within 800ns of its detection of the BSY signal.

Figure 2: Bus Phases for INQUIRY Command

1.4 SCSI Bus signals

The SCSI bus uses eighteen signals. Nine are control signals used to develop logical bus phases, and nine are data signals, including parity, for messages, commands, status, and data.

The state of the SEL, BSY, and I/O signals and the sequence of the phases determine when the Bus Free, Arbitration, Selection, and Reselection phases are entered.

Table 1 describes the nine control signals.

- The Selection or Reselection phase can be entered onl form the Arbitration phase.
- The Arbitration phase can be entered only from the Bus Free phase.
- The Bus Free phase can be entered from any of the other phases (although some transitions are caused by errors).

'I'o determine the current phase, you need to know information about the previous phase and the state of the signals. The initiator and target drive these signals to change from one phase to another phase.

Table 1: SCSI Bus Signals

Signal	Description
BSY(BUSY)	An 'OR-tied signal which indicates that the bus is being used.
SEL(SELECT)	A signal used by an initiator to select a target, or by a target to reselect an initiator.
C/D (CONTROL/DATA)	A signal drven by a target to indicate whether or not control or data information is on the data bus. True indicates control.
I/0 (INPUT/OUTPUT)	A signal driven by a target to control the direction of data movement on the data bus. True indicates input to the initiator. This signal is also used to distinguish between selection and reselection phases.
MSG (MESSAGE)	A signal driven by a target during the Message phase.
REQ (REQUEST)	A signal driven by a target to request a REO/ACK data transfer handshake.
ACK (ACKNOWLEDGE)	A signal driven by an iniüator to acknowledge a REO/ACK data transfer.
ATN (ATTENTION)	A signal driven by an inifiator to indicate the Atterifion conditioti (inidator has a message for the target).
RST (RESET)	An 'OR-tied' signal and hard Res.et condition.
DB (7-0,P) (DATA BUS)	Eight data-bit(DB) signals, plus a parity-bit signal that form a Data Bus. DB(7) is the most significant bit and has the highest phohty during the Arbitradon phase. Bit number, significance, and priority decreaso downward to DB (0). A data bit is defined as one when the signal value is true, and defined as zero when the signal value is false. Data parity DB(P) shall be odd, but parity is undefined during the Arbitradon phase.

NOTE

For the measurements which follow, SCSI bus termination is assumed to be external to the SCSI device. SCSI deviees may have the provision of allowing optional internal termination.

1.4.1 Single-Ended Cable:

All assigned signals are terminated with 220 ohms to +5 volts (nominal) and 330 ohms to ground at each end of the cable.

All signals use open-collector or three-state drivers.

1.4.2 Output Characteristics:

Each signal driven by a SCSI device has the following output characteristics when measured at the SCSI device's connector:

Signal assertion 0.0 volts dc to 0.5 volts dc at 48 milliamps (sinking).
Signal negation 2.5 volts dc to 5.25 volts dc.

1.4.3 Input Characteristics:

SCSI device signal receivers have the following input charactelistics:

Signal true = 0.0 volts dc to 0.8 volts dc
Maximum total input load = 0.4 milliamps at 0.5 volts dc
Signal false = 2.0 volts dc to 5.25 volts dc
Minimum input hysteresis = 0.2 volts dc

1.4.4 Termination Power:

SCSI initiators supply terminator power to the Termpwr pin(s). This power supplied through a diode or similar semiconductor that prevents backflow of power to the SCSI device. Any SCSI device may supply terminator power.

All terminators independent of location are powered from the Termpwr pin(s).

1.4.5 SCSI Bus:

Communication on the SCSI bus is allowed between only two SCSI devices at any given time. There is a maximum of eight SCSI devices. Each SCSI device has a SCSI ID bit assigned. When two SCSI devices communicate on the SCSI bus:

One acts as an initiator and the other acts as a target.

The initiator originates an operation and the target performs the operation.

A SCSI device usually has a fixed role as an initiator or target, but some devices are able to assume either role. In most cases, the host is the initiator and the device is the target.

An initiator may address up to eight peripheral devices that are connected to a target. These are called Logical Units Numbers (LUNs). Digital devices currently only support a single LUN per deviee.

Figure 3: SCSI ID Bits

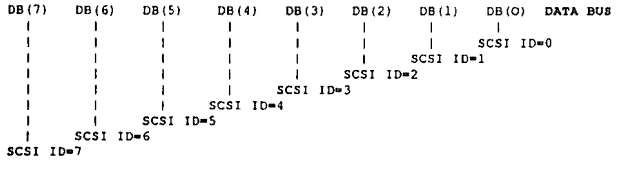


Fig. 3 SCSI ID Bits

1.4.6 Figure SCSI Configurations

Up to eight SCSI devices can be supported on the SCSI bus. They ean be any combination of initiators and targets. Certain SCSI bus functions are assi,-ned to the initiator and certain SCSI bus functions are assigned to the target. The initiator arbitrates for the

SCSI bus and seleets a particular target. The target requests the transfer of Cornmand, Data, Status, or other information on the Data Bus, and in some cases it arbitrates for the SCSI bus and reselects an initiator for the purpose of continuing an operation.

Information transfers on the Data Bus are asynchronous and follow a defined REQ/ACK handshake protocol. One byte of information may be transferred with each handshake. An option is defined for synchronous data transfer.

1.4.7 Signal Values

Signals may assume true or false values. There are two methods of di-iving these signals. In both cases, tlie signal shall be actively driven true, or asserted. In the case of OR-tied drivers, the diiver does not drive the signal to the false state, rather the bias circuitry of the bus terminators pulls the signal false whenever it is released by the drivers at every SCSI device. If any driver is asserted, then the signal is true.

In the case of non-OR-tied drivers, the sigrial may be actively driven false, or negated. The advantage to actively driving signals false during the information transfcr is that the transition from true to false occurs more quickly, and the noise margin is much higher than if the signal is simply released; this is required to reliably transfer data at maximum rates.

1.4.8 OR-Tied Signals

The BSY and RST signals are OR-tied only. In the ordinary operation of the bus, these signals may be simultaneously driven true by several drivers. No signals other than BSY, RST and DB(P) are simultaneously driven by two or more drivers, and any signal other than BSY and RST may employ OR-tied or non-OR-tied drivers. DB(P) is not driven false during the Arbitration phase.

Table 2: Signal Sources

A Cable Signals C/D, I/O

Bus Phase	BSY	SEL	REQ	ATN	MSG, DB(P)	ACK,	DB(7.0)
BUS FREE	None	None	None	None	None		
ARBITRATION	All	Win	None	None	S ID		
SELECTION	I&T	Init	None	Init	Init		
RESELECTION	I&T	Targ	Targ	Init	Targ		
COMMAND	Targ	None	Targ	Init	Init		
DATA IN	Targ	None	Targ	Init	Targ		
DATA OUT	Targ	None	Targ	Init	Init		
STATUS	Targ	None	Targ	Init	Targ		
MESSAGE IN	Targ	None	Targ	Init	Targ		
MESSAGE OUT	Targ	None	Targ	Init	Init		

All: The signal is driven by all SCSI devices that are actively arbitrating.

S ID: A unique data bit (the SCSI ID) is driven by each SCSI device that is actively arbitrating; the other seven data bits are released (i.e., not driven) by this SCSI device. The parity bit (DB(P» may be undriven or driven to ihe true state, but is never driven to ihe 'talse state during this phase.

I&T. The signal shall be driven by the initiator, target, or both, as specified in the Selection phase and Reselection phase.

Init: If driven, this signal shall be driven only by the active initiator.

None: The signal shall be released; that is, not be driven by any SCSI device. The bias circuitry of the bus terminators pulls the signal to the false state.

1.4.9 Cables

SCSI supports cable lengths of up to 6 metres, and this should be adhered to. A brief overview of SCSI development is necessary here, because many problems are caused by bad termination and cabling arrangements.

1.4.10 Development-the pitfalls

1.4.10.1 Compatibility

Because SCSI is an ANSI standard, many people believe that SCSI provides compatibility across all SCSI device types from all SCSI manufacturers, when in fact nothing could be further from the truth. The spec is ambiguous , complex and specifie to individual device types and vendors.

1.4.10.2 Electrical Integrity

When the SCSI bus was first dreamt up, some decisions were made around the parts which were cheap and readily available, rather than to the optimum electrical characteristics of the bus. Relatively low impedance iibbon cables and twisted pairs were chosen. The best off-the-shelf dliver/ receivers were designed for I/O BACKPLANE drivers, resulting in a drive current of 48mA. Terminators were chosen to match the drivers, and stub lengths were designed to handle most known PD topologies.

The result of these convenient choices is that the cable is incorrectly terminated. Stubs, receivers, terminators and connectors all contain some mismatch, and further amplify the problem. The result is that every time a signal transition takes place, ringing is observed on the bus. The uncontrolled rise-times of the drivers add to the problem by adding considerable high frequency content to the transition, and of course the faster the repetition the less likely the bus will settle correctly before the next transition.

This phenornenon creates the situation where the larger the number of devices, the longer the stubs, the longer the cable and the faster the bus is operated, the higlier the probability of creating an error during transmission. The problem is so severe, that glitches and false triggering have been seen with as few as 2 devices connected to the bus. Attempts to connect the full complement of 7 devices almost always results in high data error rates, or in many cases total bus hangs. The reason for this is that over the length of the bus, the signals show little or no voltage margins at some or maybe even all of the device connection points.

1.4.10.3 Common Mode Noise

Any time a cable is led from one cabinet to another, the possibility of voltage/noise dlifferences over ground lines becomes a reality. The situation is exacerbated by different power supplies, or even worse, seperate AC supplies to the expansion boxes. DECs own drives use a single ended cable, whicli means the noise immunity is limited to the signal margins of the bus. SCSI-2 does allow for the use of difterential drivers, and some discussion exists on the possibility of providingjumpers on drive /controller modules to switch between single-ended or differential drivers. However the counter arguement is that this will add to the total connector count on the bus and degrade the signal integrity even further. Currently Digital ships only devices conforming to the single-ended driver standard.

There is also a problem with trying to configure devices/controllers which support differntial signals, to those which support single-ended signals. The termination is different, but as if that isn't bad enough, the actual pinout on the connectors is also different. 'Ib glve an example, the single- ended cable has a supply to the terminator on pin 26 of +5v. The differential cable has 'polarity' across pins 25 and 26, the difference being that -ve is on pin 26. Also, to show how there would certainly be parity errors, notice that pin 2 of the single-ended cable is used for data -bit 0, yot it is GROUND on the differential cable. These anomolies in pinout

exist throughout almost every signal on the SCSI bus, the result being that it just doesn't work at all. These many problems with the poor electrical design of the cable has meant that very few systems can be fully configured. So although the ANSI SCSI spec allows a logical connection of up to 8 'nodes', this is never practical in the electrical environment in which these devices reside. The result is that we see many systems restricted to 4 devices per SCSI controller. Controllers such as the KZQSA do not support disk drives, and they only support a maximum of 4 tape drives. Connecting disks onto the controllers often works, but tests with such configurations have always shown high error rates and so the KZQSA is marketed as a tape controller.

CHAPTER 2

LOGICAL CHARACTERISTICS

This chapter provides an overview of the eight SCSI Bus Phases. It also contains relevant SCSI Bus information related to these eight Phases.

2.1 SCSI Bus Phases

The SCSI architecture includes eight distinct phases:

Bus Free phase
Arbitration phase
Selection phase
Reselection phase.

Command phase > These phases are collectively
Data phase > termed the Information Transfer
Message phase > Status phase Phases

The SCSI bus can never be in more than one phase at any given time. Unless otherwise noted in the following description, signals that are not mentioned shall not be asserted.

2.1.1 Bus Free Phase

The Bus Free Phase is used to indicate that no SCSI device is actively using the SCSI bus and that it is available for subsequent users. SCSI devices shall detect the Bus Free Phase after SEL and BSY are both false for at least a bus settle delay. SCSI devices shall release all SCSI bus signals hin a bus clear delay after BSY and SEI, become continuously false fbr a bus settle delay. If a SCSI device requires more than a bus settle delay to detect the Bus Free Phase then it releases all SCSI bus signals within a bus clear delay minus the excess time to detect the Bus Free Phase. The total time to clear the SCSI bus does not exceed a bus settle delay plus a bus clear delay. Initiators normally do not expect Bus Free Phase to begin because of the target's release of BSY except after one of the following occurrences:

* After a Reset condition is detected

* After an Abort message is successfully received by a target

* After a Bus Device Reset message is successfully received by a target

* After a Disconnect message is successfully transmitted from a target

* After a Command Complete message is successfully transmitted from a target

* Aftei- a Release Recovery message is successfully received by a target.

The Bus Free Phase may also be entered after an unsuccessful selection or reselection, of SEL rather than the relase of BSY that first allhough in this case it is the established the Bus Free Phase.

If an initiator detects the release of BSY by the target at other times, the target is indicating an error condition to the initiator. The target may perform this transition to the Bus Free Phase independent of the state of the ATN signal. The initiator manages this condition as an unsuccessful I/0 process termination. The target terminates the I/0 process by clearing all pending data and status information for the affected logical unit or process. The target may optionally prepare sense bytes that could be read by a Request Sense command. When an initiator detects an unexpected Bus Free Kondition it is normal that a Request Sense command is attempted to obtain any valid sense information that may be available. If the error that caused the Bus Free terminatiori of the 1/0 process is still present, the Request Sense command may not be successful.

2.1.2 Arbitration Phase

The Arbitration Phase allows one SCSI device to gain control of the SCSI bus so that it can assume the role of an initiator or target. The procedure for an SCSI device to obtain control of the SCSI bus is as follows:

1. The SCSI device shall first wait for the Bus Free Phase to occur. The Bus Free Phase is detected whenever both BSY and SEL are simultaneously and continuously false for a minimum of a bus settle delay

NOTE:

This bus settle delay is necessary because a transmission line phenomenon known as a "wire-OR glitch" may cause BSY to briefly appear false, even though it is being driven true.

2. The SCSI device waits a minimum of a bus free delay alter detection of the Bus Free Phase (that is after BSY and SEL are both false for a bus settle delay) before driving any signal

3. Following the bus free delay in Step (2), the SCSI device may arbitrate for the SCSI bus by asserting both BSY and its own SCST ID, however the SCSI device does not arbitrate (that is assert BSY and its SCSI ID) if more than a bus settle delay has passed since the Bus Free Phase was last observed

4. After waiting at least an arbitration delay (measured from its assertion of BSY) the SCSI device examines the Data Bus 1f a higher **priority** SCSI ID bit is true on the DATA BUS (DB(7) is the highest), then the SCSI device has lost the arbitration and the SCSI device releases its'signals and returns to Step (1)' If no higher priority SCSI ID bit is true on the Data Bus, then the SCSI device has won the arbitration and it asserts SEL. Any other SCSI device that is participating in the Arbitration Phase has lost the arbitration and releases BSY and its SCSI ID bit within a bus clear delay after SEL becomes true. A SCSI device that loses arbitration returns to step (1)

5. The SCSI device that wins arbitration shall wait at least a bus clear delay plus a bus settle delay after asserting SEL before changing any signals.

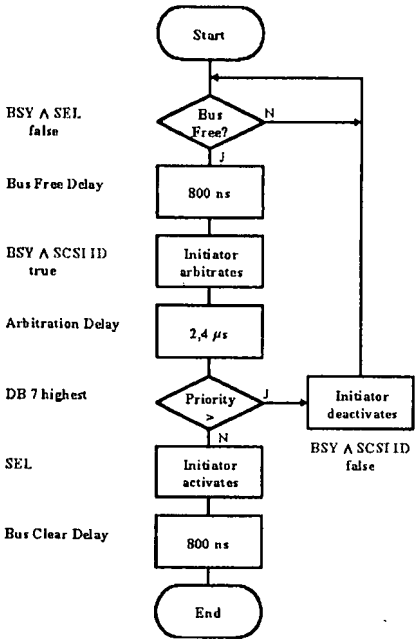


Fig. 3.1 SCSI Arbitration

NOTE:

The SCSI ID bit is a single bit on the Data Bus that corresponds to the SCSI device's unique SCSI address. All other seven Data Bus bits are released by the SCSI device. Parity is not valid during the Arbitration Phase. During the Arbitration Phase, DB(P) may be released or asserted, but is not actively driven false.

2.1.3 Selection Phase

The Selection Phase allows an initiator to select a target for the purpose of initiating some target function for example the Read or Write command.

NOTE:

During the Selection Phase the I/O signal is negated so that this phase can be distinguished from the Reselection Phase.

The SCSI device that won the arbitration has both BSY and SEL asserted and has delayed at least a bus clear delay plus a bus settle delay before ending the Arbitration Phase. The SCSI device that won the arbitration becomes an initiator by not asserting the I/O signal.

The initiator sets the Data Bus to a value which is the OR of its SCSI ID bit and the target's SCSI ID bit. The initiator then waits at least two deskew delays and releases BSY. The initiator then waits at least a bus settle delay before looking for a response from the target.

Figure 4: Selection Phase

DB(0)	DB(1)	DB(2)	DB(3)	DB(4)	DB(5)	DB(6)	DB(7)
1	0	0	0	0	0	1	0

In this example, SCSI ID 6 (host) has asserted both it's own SCSI ID (DB(6)) and that of a device (DB(0)).

The target determines that it is selected when SEL and it's SCSI ID bit are true and BSY and I/0 are false for at least a bus settle delay. The selected targtt examines the r)ata Bus in order to determine the SCSI ID of the selecting initiator. The sclected target th ('.n ;-ts,-,erts BSY within a selection; this is required for correct operation of the time-out procedure.

The target shall not respond to a selection if bad parity is detected. Also, if more than two SCSI ID bits are on the Data Bus, the target does not respond to selection.

2.1.4 Reselection Phase

Reselection is an optional phase that allows a target to reconnect to an initiator for the purpose of continuing some operation that was previously started by the initiator but was suspended by the target. For example, a host system may have requested a Read from a disk. The disk can Disconnect and Reconnect if the Read involves a time consuming seek operation to be performed. This is one of the optimization features of SCSI.

2.1.4.1 Reselection

Upon completing the Arbitration Phase, the winning SCSI device has both BSY and SEL asserted and has delayed at least a bus clear delay plus a bus settle delay. The winning SCSI device becomes a target by asserting the I/O signal. The winning SCSI device also sets the Data Bus to a value that is the OR of it is SCSI ID bit and the initiator's SCSI ID bit. The target waits at least two deskew delays and releases BSY. The target then waits at least a bus settle delay before looking for a response from the initiator.

The initiator determines that it is reselected when SEL, I/O and its SCSI ID bit are true and BSY is false for at least a bus settle delay. The reselected initiator may examine the Data Bus in order to determine the SCSI ID of the reselecting target. The reselected initiator then asserts BSY within a selection abort time of its most recent detection of being reselected; this is required for correct operation of the time-out procedure. The initiator does not respond to a Reselection Phase if bad partiy is detected. Also, the initiator may not respond to a Reselection Phase if'other than two SCSI ID bits are on the Data Bus.

After the target detects BSY, it also asserts BSY and wait at least two deskew delays and then release SEL. The target may then change

the I/0 signal and the Data Bus. After the reselected initiator detects SEL false, it releases BSY. The target continues asserting BSY until it relinquishes the SCSI bus.

2.1.5 Information Transfer Phases

The Command, Data, Status, and Message Phases are all grouped together as the Information Tansfer Phases because they are all used to transfer data or control infol-mation via the Data Bus. The actual content of the information is the scope of this section.

The C/D, I/O, and MSG signals are used to distinguish between the different Information Tansfer Phases. The target drives these three signals and therefore controls all changes from one phase to another. The initiator can request a Message Out Phase by asserting ATN, while the target ean cause the Bus Free Phase by releasing MSG, C/D, I/0 and BSY. The Information Tansfer Phases use one or more REQ/ACK handshakes to control the information transfer. Each REQ/ACK handshake allows the transfer of one byte of information. During the information transfer phases BSY remains true and SEL remains false. Additionally, during the Information Tansfer Phases, the target shall continuously envelope the REQ/ACK handshake(s) with C/D, I/0 and MSG in such a manner that these control signals are valid for a bus settle delay before the assertion of REQ of the first handshake and remain valid until after the negation of ACK at the end of the handshake of the last transfer of the phase.

Figure 5: Information Transfer Phases

MSG	C/D	I/O	Signal Phase Name	Direction of Transfer		Comment
0	0	0	Data Out	Initiator To Target	\	Data
0	0	1	Data In	Initiator From Target	/	Phase
0	1	0	Command	Initiator To Target		
0	1	1	Status	Initiator From Target		
1	0	0	*			
1	0	1	*			
1	1	0	Message Out	Initiator TO Target	\	Message
1	1	1	Message In	Initiator From Target	/	Phase

Key: 0 = False
1 = True
* = Reserved for future specitication

2.1.5.1 Asynchronous Information Transfer

The target controls the direction of information transfer by means of the I/0 signal. When I/0 is true, information is transferred from the target to the initiator. When I/0 is false, information is transferred from the initiator to the target.

If I/O is true (transfer to the initiator), the target first drives DB (7-0,P) to their desired values, delays at least one deskew delay plus a cable skew delay, then assert REQ. DB(7-0,P) remains valid until ACK is true at the target. The initiator reads DB(7-0,P) after REQ is true, then signals its acceptance of the data by asserting ACK. When ACK becomes true at the target, the target may change or release DB(7-0,P) and negate REQ. After REQ is false the initiator then negates ACK. After ACK is false the target may continue the transfer by driving DB(7-0,P) and asserting REQ, as described above.

If I/0 is false (transfer to the target) the target requests information by asserting REQ. The initiator drives DB(7-0,P) to their desired values, delays at least one deskew delay plus a cable skew delay and asserts ACK. The initiator continues to drive DB(7-0,P) until REQ is false. When ACK becomes true at the target, the target reads DB(7-0,P) then negates REQ. When REQ becomes false at the initiator, the initiator may change or release DB(7-0,P) and negates ACK. The target may continue the transfer by asserting REQ, as described above.

2.1.5.2 Synchronous Data Transfer

Synchronous data transfer is only used in data phases. lt is used in a data phase if a synchronous data transfer agreement has been established between the target and initiator.

The REQ/ACK offset specifies the Maximum number of REQ pulses that can be sent by the, target in advance of the number of ACK pulses received from the initiator, establishing a pacing mechanism. If the number of REQ pulses exceeds the number of ACK.pulses by the REQ/ACK offset, the target shall not assert REQ until alter the leading edge of the next ACK pulse is received. A requirement for successful conipletion of' the data phase is that the number of ACK and REQ pulses be equal.

The target asserts the REQ signal for a minimum of an assertion period. The target shall wait at least a transfer period from the last transition of REQ to false before asserting the REQ signal. The initiator sends one pulse on the ACK signal for each REQ pulse received. The ACK signal may be asserted as soon as the leading edge of the corresponding REQ pulse has been received. The initiator asserts the ACK signal for a minimum of an assertion period. The initiator waits at least the greater of a transfer period from the last transition of ACK to true or for a minimum of a negation period from the last transition of ACK to false before asserting the ACK signal.

If I/0 is true (transfer to the initiator), the target first drives DB(7-0,P) to their desired values, waits at least one deskskew delay plus one cable skew delay, then assert REQ. DB(70,P) shall be held valid for a minimum of one deskew delay plus one cable skew delay plus one hold time after the assertion of REQ. The target asserts REQ for a minimum of an assertion period. The target may then negate REQ and change or release DB(7-0,P). The initiator reads the value on DB(7-0,P) within one hold tirne of thc transition of REQ to true. The initiator then responds with an ACK pulse.

If I/0 is false (transfer to the target), the initiator transfers one hyte for each REQ pulse received. After recciving the leading edge of a REQ pulse, the initiator first drives DB (7-0,P) to their desired values, delays at least one deskew delay plu one cable skew delay, then asserts ACK. The initiator holds DB(7-0,P) valid for at least one deskew delay plus one cable skew delay plus one, hold time after the assertion or ACK. 'I'he initiator asserts ACK for a minimum of an assertion period. The initiator may then negate ACK and may change or release DB (7-0, P). The target reads the value of DB(7-0,P) within one hold time of the transition of ACK to true.

2.1.6 Command Phase

The Command Phase allows the target to request command information from the initiator.

The target shall assert the C/D signal and negate the I/0 and MSG during the REQ/ACK handshake(s) of this phase.

2.1.7 Data Phase

The Data Phase is a term that encompasses both the Data In Phase and the Data Out Phase.

2.1.7.1 Data In Phase

The Data In Phase allows the target to request that data be sent to the initiator from the target.

The target asserts the I/0 signal and negate the C/D and MSG signals during the REQ/ACK handshake(s) of this phase.

2.1.7.2 Data Out Phase

The Data Out Phase allows the target to request that data be sent from the initiator to the target.

The target negates the C/D, I/0, and MSG signals during the REQ/ACK handshake(s) of this phase.

2.1.8 Status Phase

The Status Phase allows the target to request that status information be sent from the target to the initiator.

The target asserts C/D and I/0 and negates the MSG signal during the REQ/ACK handshake of this phase.

2.1.9 Message Phase

The Message Phase is a term that references either a Message In, or a Message Out Phase. Multiple messages may be sent during either phase. The first byte transferred in either of these phases is either a single-byte message or the first byte of a multiple-byte message. Multiple-byte messages are wholly contained within a single message phase.

2.1.9.1 Message In Phase

The Message In Phase allows the target to request that message(s) be sent to the initiator from the target.

The target asserts C/D, 1/0, and MSG during the REQ/ACK handshake(s) of this phase.

2.1.9.2 Message Out Phase

The Message Out Phase allows the target to request tliat message(s) be sent from the initiator to the target. The target invokes this phase in response to the Attention condition created by the initiator.

The target asserts C/D and MSG and negate I/0 during the REQ/ACK handshake(s) of this phase. The target haiidshakes byte(s) in this phase until ATN is negated, except when rgiecting a message.

If the target detects one or more parity erroks) on the message byte(s) received, it may indicate its desire to retry the message(s) by asserting REQ after detecting ATN has gone false and prior to changing to any other phase. The initiator, upon detecting this condition, shall be send all of the previous message byte(s) in the same order as previously sent during this phase. When re-sending more than one message byte, the initiator shall assert ATN prior to asserting ACK on the first byte and shall maintain ATN asserted until the last byte is sent.

The target may act on messages as received as long as no parity error is detected and may ignore all remaining messages sent under one ATN condition after a partiy error is detected. Whcn a sequence of messages is re-sent by an initiator because of a target detected parity error, the target shall not act on any message which it acted on the first time reccived.

If the target receives all of the message byte(s) successfully, that is no parity errors, it shall indicate that it does not wish to retry by changing to any information transfer phase other than the Message Out Phase and transfer at least one byte. The target may also indicate that it has successfully received the message byte(s) by changing to the Bus Free Phase for example the Abort or Bus Device Reset messages).

2.2 SCSI Bus Conditions

The SCSI bus has two asynchronous conditions; the Attention condition and the Reset condition. These conditions cause the SCSI device to perform certain actions and can alter the phase sequence.

2.2.1 Attention Condition

The Attention condition allows an initiator to inform a target that the initiator has a message ready. The target may get this message by performing a Message Out Phase.

The initiator creates the Attention condition by asserting ATN at any time except during the Arbitration or Bus Free Phases.

The initiator asserts the ATN signal before negating ACK for the last byte transferred in a bus phase for the Attention Kondition to be honored before transition to a new bus phase. An ATN asscrted later might not be honored until a later bus phase and then may not result in the expected action. The initiator negates ATN before asserting ACK when transferring the last byte of the messages. If the target detects that the initiator failed to meet this requirement, then the target goes to Bus Free Phase (unexpected Bus Free). A target shall re,spond witli Message Out Phase as follows:

- If ATN occurs during a Command Phase, Message Out occurs -.ifter transfer of part or all command descriptor block bytes have been completed
- If ATN occurs during a Data Phase, Message Out occurs at the target's convenience (not necessarily on a logical block boundary). The initiator must continue REQ/ACK handshakes until it detects the phase change
- If ATN occurs during a Status Phase, Message Out occurs after the staus byte has been acknowledged by the initiator
- If ATN occurs during a Message In Phase, Message Out Phase orriirs after the current Message In byte has been acknowledged by the initiator, but before tlie target asserts the REQ for any following Message In byte. This permits a Message Parity Error message from the initiator to be associated with the appropriate message byte
- If ATN occurs during a Selectioti Phase and before the initiatot- releascs the BSY signal, Message Out occurs immediately after that Selection phase
- If ATN occurs during a Reselction Phase, Message Out occurs after the target has sent its Identify message for that Reselection Phase. The initiator keeps ATN asserted if more than one byte is to be transferred. The initiator may negate the ATN signal at any time except it does not negate the ATN signal while the ACK signal is asserted during a Message Out Phase. Normally, the initiator negates ATN while REQ is true and ACK is false during the last REQ/ACK handshake of the Message Out Phase.

2.2.2 Reset Condition

The Reset Kondition is used to immediately clear all SCSI devices from the bus. This condition takes precedence over all other phases and conditions. Any SCSI device may create the Reset Kondition by asserting RST for a minimum of a reset hold time During the Reset Kondition the state of all SCSI bus signals other than RST is not defined.

All SCSI devices release all SCSI bus signals (except RST) within a bus clear delay of the transition of RST to true. The Bus Free Phase always follows the Reset Kondition

Hard Reset Alternative

SCSI devices are required to implement "Hard Reset" as defined in the ANSI standard. The device, upon detection of the Reset Kondition

1. Clears all I/0 processes including queued YO processes

2. Releases all SCSI device reservations

3. Returns any SCSI device operating modes to their appropriate initial conditions, similar to those conditions that would be found after an normal power-on reset. Mode Select conditions are restored to their last saved values if saved values have been established. Mode Select conditions for which no values have been saved are returned to their default values

4. Unit Attention Kondition is set.

Soft Reset Alternative

This alternative is not currently supported.

2.3 SCSI Bus Phase Sequences

The order in which phases are used on the SCSI bus follows a prescribed sequence.

The Reset condition can abort any phase and is always followed by the Bus Free Phase. Also any other phase can be followed by the Bus Free Phase but many such instances are error conditions. (See Section 2.1.1).

The normal Progression is from the Bus Free Phase to Arbitration, from Arbitration to Selection or Reselection, and from Selection or Reselection to one or more of the Information Transfer Phases (Command, Data, Status, or Message). Normally, the final Information Humanster Phase is Message In Phase where a Disconnect, Command Complete, or Linked Command Complete message is transferred, followed by Bus Free Phase.

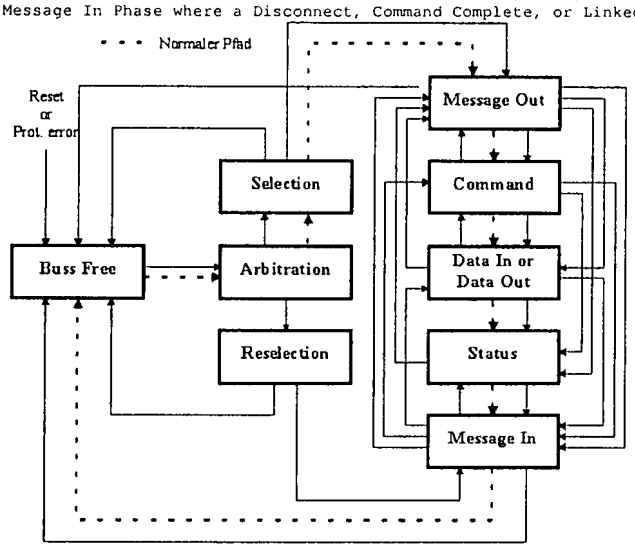


Fig. 6 SCSI Phase Sequences

2.4 SCSI Pointers

Consider the system shown in Figure 7 in which an initiator and target communicate on the SCSI bus in order to execute an I/O operation.

Figure 7: Simplified I/O Operation

The SCSI architecture provides for two sets of three pointers within each initialor. The pointers reside in the initiator SCSI bus control section (often referred to as a host adapter). The first set of pointers are known as the current (or active) pointers. These pointers are used to represent the state of the interface and point to the next command, data, or status byte to be transferred between the initiator's memory and the target. There is only one set of current pointers in each initiator. The current pointers are used by the target currently connected to the initiator.

The second set of pointers are known as the saved pointers. There is one set of saved pointers for each I/0 process that is currently active (whetlier or not it is currently coniieccod). Th,e saved command pointer always four points to the start of the command descriptor block for the current I/0 process. The saved status pointer always points to the start of the status area for the current PO process. At the beginning of each I/0 process, the saved data pointer points to the start of the data area. lt remains at this value until the tarc,'et sends a Save Data Pointer message to the in'itiator. In response to this message, the initiator stores the value of the current data pointer into the saved data pointer. The Target may restore the current pointers to their saved values by sending a Restore Poiiiters message to the initiator. The initiator then moves the saved value of each pointer into the corresponding current pointer. Whenever an SCSI device disconnects from the bus, only the saved pointer values are retained. The current pointer values are restored from the saved values upon the next reconnection.

2.5 Message Systern Description

The message system allows communication between an initiator and target for the purpose of interface management. A message may be one, two, or multiple bytes in length. One or more messages may be sent during a single Message Phase, but a message may not be split over Message Phases. The initiator is required to end the Message Out Phase (by negating ATN) when it sends certain messages identified in Figure 8.

One byte, two byte and extended message formats are defined. The first byte of the message determines the format as follows:

Figure 8: Message Values

Value	Message Format
00h	One-Byte Message (Command Complete)

01h	Extended Messages
02h - 1 Fh	One-Byte Messages
20h - 2Fh	Two-Byte Messages
30h - 7Fh	Reserved
80h - FFh	One-Byte Message (Identify)

One-byte messages consist of a single byte transferred duling a Message phase. The value of the byte determines which message is to be performed as defined in Figure 9.

'Avo-byte messages consist of two consecutive bytes transferred during a Message Phase. The value of the first byte determines which message is to be performed as defined in the next figure. The second byte is a parameter byte which is used as defined in the message description. A value of one in the first byte of a message indicates the beginning of a multiple-byte extended message. The minimum number of bytes sent for an extended message is three.

Figure 9: Message Codes

Code	Support Init/Targ	Message before Last ACK	Name	Direction	Negate ATN					
06h	O	M	Abort	Out	Yes					
23h	O	O	Abort Tag (Two Bytes)		Out	Yes				
0Ch	O	M	Bus Device Reset		Out	Yes				
0Eh	O	O	Clear Queue	Out	Yes	00h	M	M	Command Complete	In	---
04h	M	M	Disconnect	In	---					
04h	O	O	Disconnect	Out	Yes					
...	O	O	Extended Identify	In	Out	No				
80h&	M	O	Indentity	In	---					
80h&	M	M	Identify	Out	No					
24h	O	O	Ignore Wide Residue (Two Bytes)		In		---			
0Fh	O	O	Initiate Recovery	In	Out	Yes				
05h	M	M	Initiator Detected Error			Out	Yes			
0Ah	O	O	Linked Command Complete	In		---				
0Bh	O	O	Linked Command Complete (With Flag)			In	---			
09h	M	M	Message Parity Error			Out	Yes			
07h	M	M	Message Reject In		Out	Yes				
...	O	O	Modity Date Pointer		In	---				
08h	M	M	No Operation		Out	Yes				
			Queue Tag Messages (Two Bytes)							
2 h	O	O	Head Of Queue Tag		Out	No				
20h	O	O	Mundane Queue Tag	In	Out	No				
22h	O	O	Ordored Queue Tag		Out	No				
10h	O	O	Release Recovery		Out	Yes				
03h	M	M	Restore Pointers	In		---				
02h	M	M	Save Date Pointer	In		---				
***	M	M	Synchronous Data Transfer Request			In	Out	Yes		
***	O	O	Wide Dita Transfer Request		In	Out	Yes			
0Dh			Reserved							
11h -	1Fh		Reserved							
25h -	2Fh		Reserved for two-byte messages							
30h -	7Fh		Reserved							

Key:

M =	Mandatory support
O =	Optional support
In =	Target to initiator
Out	Initiator to target
Yes	Initiatorshall negate ATN before last ACK of message
No =	Initiator may or may not negate ACK before last ACK of message (see Section 2.2.1 Not applicable

*** = Extended message

80h& = Codes 80h through FFh are used for IDENTIFY messages

Table 3: Extended Message Format

Byte	Value	Description
0	01h	Extended message
1	n	Extended message length
2	y	Extended message code
3 -n + 1	x	Extended message arguments

The extended message length specifies the length in bytes of the extended message code plus the extended message arguments to follow. Therefore, the total length of the message is equal to the extended message length plus two. A value of zero for the extended message length indicates 256 bytes follow.

The extended message codes are listed in Table 4. The extended message arguments are specified within the exte nded message descriptions.

The first message sent by the initiator after the Selection Phase shall be the Identify, Abort, or Bus Device Reset message. If a target receives any other message it shall go to Bus Free Phase (unexpected Bus Free).

If the first message is Identify message, then it may be imme(Iiately followed by other messages, such as the first of a pair of Synchronous Data Uansfer Request messages. The Identify message establishes a logical connection between the initiator and the specified logic-al unit or target routine within the target known as an I - T - L nexus or I_@R nexus. After the Reslection Phase, the target's first message shall be Identify. This allows the I - T - L nexus or I - T - R nexus to be re-established. Only one logi@ unit or target routine shall be identified for any connection; a second Identify message with a different lo,-ical unit number or target routine number shall not be sent before the SCSI bus has been released (Bus Free Phase).

All initiators implement the mandatory messages tabulated in the "Init" column of Figure 9. All targets shall implement the mandatory messages tabulated in the "Targ" column of Figure 9 Whenever an I - T - L nexus or I-T-R nexus is estabilished by an initiator that is allowing disconnection, the initiator ensures that the active pointers are equal to the saved pointers for that particular logical unit or target routine. An implied restore pointers operation occurs as a result of a reconnection.

Table 4: Extended Message Codes

Code	(y)	Description
02h		Extended Identify
00h		Modify Data Pointer
01h		Synchronous Data Transfer Request
03h		Wide Data Transfer Request
04h	7Fh	Reserved
80h	FFh	Vendor Unique

2.6 Messages

The SCSI messages are defined in this section.

2.6.1 Abort

This message is sent from the initiator to the target to clear the present I/0 process plus any queued I/0 process for the I-T-x nexus. The target shall go to the Bus Free Phase following successful receipt of this message. Pending data, status, and queued I/0 processes for any other. I-T-x nexus shall not be cleared. If only an I-T nexus has been established, the target shall go to the Bus Free phase. No status or message is sent for the I/0 process and the I/0 process queue is not affected. It is not an error to issue this message to an l_@x nexus that does not currently have an active or queued I/0 process. @ansmission of this message terminates any extended Kontingent allegiance condition that may exist between the I-T-x nexus.

2.6.2 Bus Device Reset

This message is sent from an initiator to direct a target to clear all current I/0 processes on that SCSI device. This message forces a hard reset Kondition to the selected SCSI dlive. The target goes to the Bus Free Phase following successful receipt of this message. The target creates a unit attention condition for all initiators.

2.6.3 Command Complete

This message is sent from a target to an initiator to indicate that the execution of a command has terminated and that valid status has been sent to the initiator. After successfully sending this message, the target enters the Bus Free Phase by releasing BSY. The tar,-et considers the message transmission to be successful when it detects the negation of ACK for the Command Complete message with the ATN signal false.

2.6.4 Disconnect

This message is sent from a target to inform an initiator that the present connection is going to be broken (thc target plans to disconnect by releasing BSY), but that a latcr reconnect will be required in order to complete the eurrent 1/0 process. This message does not cause the initiator to save the data pointer. After successfully sending this message, the target goes to the Bus Free Phase by releasing the BSY signal. The target considers the message transmission to be successful when it detects the negation of the ACK signal for the Disconnect message with the ATN signal false.

NOTE:

Targets which break data transfers into multiple connections should end each successful connection (except the last) with a Save Data Pointer - Disconnect message sequence. However, in the case of VMS, any deviee which issues a Disconnect will have an implied Save Data Pointers performed by the Port Driver.

This message may also be sent from an initiator to a target to instruct the target to disconnect from the SCSI bus. If this option is supported and after the Disconnect message, the target shall switch to Message In Phase, send the Disconnect message to the initiator (possibly preceded by Save Data Pointer message), and then disconnect by releasing BSY After releasing BSY, the target does not participate in another Arbitration phase for at least a disconnection delay. If this option is not supported or the target cannot disconnect at the time when it receives the Disconnect message from the initiator, the target responds by sending Message Reject message to the initiator.

2.6.5 Identify

The Identify message is sent by either the initiator or the target to establish an I - T - L nexus or an 1-T-R nexus. The logical unit number addresses one of up to eight logical devices attached to a target. There are however, no Digital devices which currently support multiple LUNS.

NOTE:

Use of the Identify message to establish an I-R nexus allows connection to one of up to eight target routines or functions in the target itself. These target routines or functions are expected to be used for maintenance and diagnostic purposes.

Figure 10: Identify Message Format

Bit Byte	7	6	5	4	3	2	1	0
0	Identify	DiscPriv	Luntar	Reserved	Reserved		Luntpn	

2.6.6 Initiator Detected Error

This message is sent from an initiator to inform a target that an error has occurred that does not preclude the target from retrying the operation. The source of the error may either be reiated to previous activities on the SCSI bus or may be internal to the initiator and unrelated to any previous SCSI bus activity. Although present pointer integrity is not assured, a Restore Pointers message or disconnect followed by a reconnect, shall cause the pointers to be restored to their defined prior state.

2.6.7 Message Parity Error

This message is sent from the initiator to the target to indicate that the last message byte it received had a parity error. In order to indicate its intentions of sending this message, the initiator asserts the ATN signal prior to its release of ACK for the REQ/ACK handshake of the message that has the parity error. This provides an interlock so that the target can determine which message has the parity error. If the target receives this message under any other circumstance, it shall signal catastrophic error condition by releasing the BSY signal without any further information transfer attempt.

2.6.8 Message Reject

This message is sent from either the initiator or target to indicate that the last message byte it reeeived was inappropriate or had not been impleiuented.

In order to indicate its intentions of sending this message, the initiator asserts the ATN signal prior to its release of ACK for the REQ/ACK handshake of the message byte that is to be rejected. If the target receives this message under any other circumstance, it rejects this message.

When a target sends this message, it changes to Message In Phase and sends this message prior to requesting additional message bytes from the initiator. This provides an interlock so that the initiator can determine which message byte is rejected.

2.6.9 No Operation

This message is sent from an initiator in response to a target's request for a message when the initiator does not currently have any other valid message to send.

2.6.10 Restore Pointers

This message is sent from a target to direct the initiator to restore the most recently saved pointers (for the cux.-i.-ent nexus) to the active stale. Pointers to the command- data, and scatus locations for the nexus are restored to the active, pointers. Command and stitus pointers are restoi.-ed to the beginning of the preseiit command and status areas. The data pointer are restored to the vali-ie at the beginning of the data area in the absence of a Save Data Pointer mess,--ige or to the value at the point at which the last Save Data Pointer message occurred for that nexus.

2.6.11 Save Data Pointer

This message is sent from a tai-get to direct the initiator to save a copy of the present aetive data pointers for the current nexus.

The target sends a Save Data Pointer message before sending a Restore Pointers message for the purpose of restoring the status pointer. The target sends a Save Data Pointer message before sending a Disconnect message if data transfer has occurred for the I/0 process. Exception: If the initiator has accepted a Modify Data Pointer message from the logical unit of the 1/0 process, it is permissible for the target to omit sending the Save Data Pointer message before sending a Disconnect message following the completion of the data transfer for the I/0 process.

Table 5: Synchronous Data Transfer Request

Byte	Value	Description
0	01h	Extended message
1	03h	Extended message length
2	01h	Synchronous Data Transfer Request code
3	m	Transfer period (m times 4 nanoseconds)
4	x	REQ/ACK offset

A Synchronous Data Transfer Request (SDTR) message exchange shall be initiated by an SCSI device whenever a previously-arranged data transfer agreement may have become invalid. The agreement becomes invalid after any condition which may leave the data transfer agreement in an indeterminate state such as:

- After a hard reset condition
- After a Bus Device Reset message
- After a power cycle.

In addition, a SCSI device may initiate an SDTR niessage exchange whenever it is appropriate to negotiate a new data transfer agreenlent (eithei- synchronous or ,tsynclironous). SCSI devices that are capable of synchronous data transfers shall not respond to an SDTR message with a Message LI%,eiect message.

The SDTR message exchange establishes the permissible transfer periods and the REQ/ACK offsets for all logical units on the two devices. The transfer period is the minimum time allowed between leading edges of successive REQ pulses and of successive ACK to mcet the device requirements for successful reception of data.

The REQ/ACK offsct is the maximum number of REQ pulses allowed to be outstanding before the leading edge of ilt's corresponding ACK pulse is received at the target. This value is chosen to prevent overflow conditions in the device's reception buffer and offset counter. A REQ/ACK offset value oi. zero indicates asynchi-oi.Lous data transfer mode; a value of FFh indicates unlimited REQ/ACK offset.

The originating device (the device that sends the first of the pair of SDTR messages) sets its values according to the rules above to permit it to receive data successfully. If the responding device can also receive data successfully with these values, it returns the aame values in it.-, SDTR message. If it requires a larger transfer period, a smaller REQ/ACK offset, or both m order to receive data successfully, it substitutes values in its SDTR message as required, retunüng unchanged any value not required to be changed. Each device when transmitting data shall respect the limits set by the other's SDTR message, but it is permitted to transfer data with larger transfer periods, smaller REQ/ACK offsets, or both, than specified in the other's SDTR message. The successful completion of an exchange of SDTR messages implies an agreement as follows:

	Responding Device SDTR response	Implied Agreement
1	Non-zero REO/ACK offset	Each device transmits data with a transfer period equal to or greater than and a REQ/ACK off set equal to or less than the values received in the other device's SDTR niessage.
2.	REQ/ACK offset equal to zero	Asynchronous transfer
3.	MESSAGE REJECT message	Asynchronous transfer

If the initiator recognizes that negotiation is required, it asserts the ATN signal and sends a SDTR message to begin the negotiatiiig process. After successfully completing the Message Out Phase, the target responds with the proper SDTR niessage. lf'an abnormal conditioll prevents the target from returning an appropriate response, both devices go to asynchronous data transfer mode for data transfers between the two devices.

Following target response 1. above, the implied agreement for sytichi-oiious operation is considered to be negated by both initiator and the target if the initiator asserts gl'N and thc first message out is either Message Parity Error or Message Reject. In this case, both devices shall go to asynchronous data transfer mode for data transfers between the two devices. For the Message Parity Error case, the implied agreement is reinstated if a retransmittal of the second of the pair of messages is successfully accomplished. AI'ter a velidor-specific number of retry attempts (greater than zero), if the target receives a Message Parity Error message, it terminates the retry activity. This may be done by either changing to any other information transfer phase and transferring at least one byte of information or by goin- to the Bus Free phase. The initiat-or accepts such actioii as abortiiig and both deviees go to asynchronous data t'ransfer mode for data transfers.between the two devices.

If the target recognizes that negotiation is required, it sends an SDTR message to the initiator. Prior to releasing ACK on the last byte of the SDTR message from the target, the initiator asserts ATIN and responds with its SDTR message or with a Message Reject message. If an abnormal condition prevents the initiator from returning an appropriate response, both devices go to @isylichronous data transfcr mode for data transf("rs betwccn the two devices.

Following an initiator's responding SDTR message, an implied agreement for synchronous operation is not considered to exist until the target leaves the Message Out Phase, indicating that the target has accepted the negotiation. After a vendor-specific number of retry attempts (greater than zero), if the target has not received the initiator's responding SDTR message, it goes to the Bus Free Phase without any further infonnation transfer attempt. This indicates that a catastrophic error condition has occurred. Both devices go to asynchronous data transfer mode for data transfers between the two devices. If, following an initiator's responding SDTR message, the target shifts to Message In Phase and the first message in is Message Reject, the implied agreement is considered to be negated and both devices go to asynchronous data transfer mode for data transfers between the two devices.

The implied synchronous agreement remains in effect until a Bus Device Reset message is received, until a hard Reset condition occurs, or until one of the two SCSI devices elects to modify the agreement. The default data transfer mode is asynchronous data transfer mode. The default data transfer mode is entered at power on, after a Bus Device Reset message, or after a hard Reset condition.

CHAPTER 3

COMMANDS AND STATUS

3.1 Introduction

This chapter defines the SCSI command and status structures and gives several examples. By keeping to a minimum the functions essential to communicate via this protocol, a widt-, range of peripheral devices can operate in the same environment. Because subsets of the full architecture may be implemented, optional functions are noted.

3.2 Command Implementation Requirernents

The first byte of all SCSI commands shall contain an operation code. Three bits (bits 7 - 5) of the second byte of each SCSI command specify the logical unit if its is not specified using the Identify message. The last byte of all SCSI commands shall contain a control block.

3.3 Command Descriptor Block

A request to a peripheral device is performed by sending a command descriptor block to the target. For several commands, the request is accompanied by a list of parameters sen.t during the Data Out Phase. See the specific commands for detailed information.

The command descriptor block always has an operation code as the first byte of the command. This is followed by an optional logical unit number, command parameters (if any), and a control byte. For all commands, if there is an invalid parameter in the command descriptor block, then the target shall terminate the command without altering the medium.

3.4 Opcodes

Opcodes are divided into two sections. Bits 5 - 7 identify a particular Group of commands. Bits 0 - 4 identify a specific command. There is a possiblity, therefore, to have 8 groups of commands and each group to hold 32 commands. Most of the commands are contained within Group 0.

The group code specifies one of the following groups:

- Group 0 - six-byte commands (see Figure 11)
- Group 1 - ten-byte commands (see Figure 12)
- Group 2 - 7bn Byte Commands (See Figure 12)
- Group 3 - reserved
- Group 4 - reserved
- Group 5 - twelve-byte commands (see Figure 13)
- Group 6 - Digital Unique

0 Group 7 - Digital Unique

3.4.1 Logical Unit Number

Digital drives support only one logical unit, LUN=0. The target shall ignore the logical unit number specified within the Command Descriptor Block if an Identify message was received.

3.4.2 Logical Block Address

The logical block address begins with block zero and is contiguous up to the last logical block on the drive. A six-byte command descriptor block contains a 21-bit logical block address. The ten-byte and the twelve-byte command descriptor blocks contain 32-bit logical block addresses. Logical block addresses in additional parameter data have their length specified for each occurrence. See the specific command descriptions.

3.4.3 Transfer Length

The transfer length field specifies the amount of data to be transferred, usually the number of logical blocks. For several commands the traiisfer length indicates the requested number of bytes to be sent as defined in the command description. For these commands the transfer length field may be identified by a different name. See the following descriptions and the individual command descriptions for fi-irther information.

Commands that use one byte for transfer length allow up to 256 blocks of data to be transferred by one command. A transfer length value of 1 to 255 indicates the number of blocks that shall be transferred. A value of zero indicates 256 blocks.

Commands that use multiple bytes for transfer length allow 65,535 or greater blocks of data to be transferred by one command. In this case, a transfer length of zero indicates that no data transfer shall take place. A value of one or greater indicates the number of blocks that shall be transferred. Refer to the specific command description for further information.

3.4.4 Parameter List Length

The parameter list length is used to specify the number of bytes sent during the Data Out Phase. This field is typically used in command descriptor blocks for parameters that are sent to a target that is mode parame@,rs, dioagnostic parameters, log parameters.

3.4.5 Allocatlon Length

The allocation length field specifies Ulie inaxinium number of bytes tliat an initiator has allocated for returned data. An allocation length of zero indicates that no data shall be transferred. This condition shall not be considered as an error. The target shall terminate the Data In Phase when allocation length bytes have been transferl-ed or when all available data have been transferred to the initiator, whichever is less.

The allocation length is typically used to return sense data, for example mode data, log dat.-i, diagnostic data, to an initiator.

3.4.6 Control Byte

The control byte is the last byte of every command descriptor block. The Flag, Link, and Digital Unique fields are not supported and must be set to zero. If any of the link, flag, or Digital unique bits are set to one, the drive shall return Check Condition status with the sense key set to Illegal Request.

3.5 Status

A status byte shall be sent from the target to the initiator during the Status phase at the termination of each command.

A description of the status byte codes is shown in Figure 15

Figure 11: Status Byte Code Bit Values

Bits of Status Byte								
7	6	5	4	3	2	1	0	Status (es) Represented
R	R	0	0	0	0	0	R	Good
R	R	0	0	0	0	1	R	Check Condition
R	R	0	0	1	0	0	R	Busy
R	R	0	1	1	0	0	R	Reservation Conflict
All Other Codes								Reserved

Key: R - Reserved bit = 0

Figure 16: Status-Definitions

Status Code	Description
Good	This status indicates that the target has successfully completed the command.
Check Condition	Any error,exception,or abnormal condition that causes sense data to be set, shall cause a Check Condition status.The Request Sense comrnand should be issued following a Check Condition status,to determine the nature of the condition.
Busy	The target is busy.This status shall be returned whenever a target is unable to process the command from an otherwise acceptable initiator.The normal Initiator recovery action is to issue the command again at a later time.
Reservation	This status shall be returned whenever a SCSI device attempts Conflict to access a logical unit or an extent within a logical unit that is reserved with a conflicting reservation type for another SCSI device (see Reserve and Release Unit commands).The normal Initiator recovery action is to issue the command again at a later time.

3.6 Command Examples

The following sections give examples of typieal command processing in the SCSI environment.

3.6.1 Single Command Example

A typical operation on the SCSI bus is likely to include a single Read command to a peripheral device. This operation is described in detail starting with a request from the initiator. This example assumes that no linked commands and no malfunctions or errors occur.

The initiator lias active pointers and a set of stored pointers representing aetive disconneeted SCSI devices (an initiator without disconnect capability does not require stored pointers). the initiator sets up the active pointers for the operation requested, arbitrates for the SCSI bus, and selects the target. Once this process is completed, the target assumes control of the operation.

The target obtains the command from the initiator (in this case, a Read command). The target interprets the command and executes it. In this case, the target gets the data from the peripheral device and sends it to the initiator. At the completion of the Read command, the target sends a status byte to the initiator. To end the operation, tlie target sends a Command Complete message to the initiator.

3.6.2 Disconnect Example

In the above single command example, the length of time necessary to obtain the data may require a time-consuming physical seek. In order to improve system throughput, the target may disconnect from the initiator, thei-eby freeing the SCSI bus to allow other I/0 process to occur. The initiator must be capable of being reselected and of restoring pointers. The target must be capable of arbitration and reselection.

After the target has received the Rcad command (and has determined that there will be a delay), it disconnects by sending a Disconnect message and releasing BSY

When the data are ready to be transferred, the target reconnects to the initiator. As a result of this reconnection, the initiator restores the pointers to their most recent saved values (which, in this case, are the initial values) and the target continues (as in the single command example) to finish the operation. The initiator recognizes that the Operation is complete when Command Complete message is received.

If target wishes to disconnect after transferring part of the data (for example while crossing a cylinder boundary), it may do so by sending a Save Data Pointer message and a Disconnect message to the initiator and then disconnecting. When reconnection is completed, the current data pointer value is restored to its value immediately prior to the Save Data Pointer message. On those occasions when an error or exception Kondition occurs and the target elects to repeat the information transfer, the target may repeat the transfer by either issuing a Restore Pointers message or by disconnecting without issuing a Save Data Pointer message. When reconnection is completed, the most recent saved pointer values are restored.

Fig. 17 shows an example of the SCSI signals, its sequence and the timing.

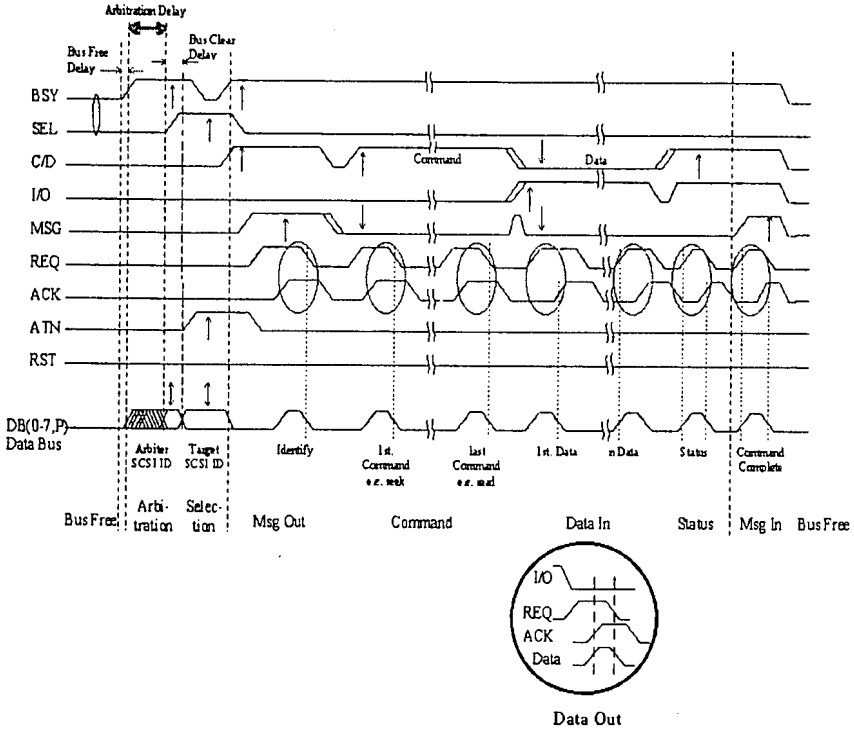


Fig. 17 SCSI Timing Diagram